

Machine Model Parameter Determination Final Report

May07-18

Client:
General Electric

Faculty Advisor:
Chen-Ching Liu

Team Members:
Jared Kline
Adam Wroblaski
Mark Reisinger
Yu Chan

Disclaimer

This document was developed as a part of the requirements of an electrical and computer engineering course at Iowa State University, Ames, Iowa. This document does not constitute a professional engineering design or a professional land surveying document. Although the information is intended to be accurate, the associated students, faculty, and Iowa State University make no claims, promises, or guarantees about the accuracy, completeness, quality, or adequacy of the information. The user of this document shall ensure that any such use does not violate any laws with regard to professional licensing and certification requirements. This use includes any work resulting from this student-prepared document that is required to be under the responsible charge of a licensed engineer or surveyor. This document is copyrighted by the students who produced this document and the associated faculty advisors. No part may be reproduced without the written permission of the senior design course coordinator.

Submitted
May 2nd, 2007

Table of Contents

Machine Model Parameter Determination Final Report.....	1
1 Introduction.....	1
1.1 Executive Summary	1
1.1.1 Project Activities.....	1
1.1.2 Final Results.....	1
1.1.3 Recommendations for Additional Work.....	1
1.2 Acknowledgements.....	1
1.3 Problem Statement.....	2
1.3.1 General Problem Statement	2
1.3.2 General Solution Approach.....	2
1.4 Operating Environment.....	2
1.5 Intended Users and Uses.....	3
1.5.1 Intended User(s).....	3
1.5.2 Intended Use(s)	3
1.6 Assumptions and Limitations	3
1.6.1 Initial Assumptions	3
1.6.2 Initial Limitations.....	3
1.7 Expected End Product and Other Deliverables.....	4
1.7.1 Simulink Block Diagram Files.....	4
1.7.3 Graphical User Interface	4
1.7.4 Project Documentation.....	4
2 Approach and Results	5
2.1 Approach.....	5
2.1.1 Functional Requirements	5
2.1.2 Design Constraints	6
2.1.3 Technical Approach Considerations	7
2.1.4 Implementation Process Description	8
2.1.5 End Product Testing.....	8
2.2 Project End Result.....	26
3 Detailed Design.....	27
3.1 Design Overview	27

3.1.1	Program Components.....	27
3.1.2	The Matlab Workspace	30
3.1.3	Setting Files	30
3.1.4	Test Data	30
3.1.5	Interaction Between Components and Flow of Information.....	31
3.2	Graphical User Interface	31
3.2.1	Data Structures.....	32
3.2.2	Opening Window	34
3.2.3	Home Window	36
3.2.4	Batch Simulation Window	44
3.3	Simulation Scripts.....	44
3.4	Simulink Models.....	44
4	Resources and Schedules	46
4.1	Resource Requirements	46
4.1.1	Personnel Effort Requirements	46
4.1.2	Other Resource Requirements	49
4.1.3	Financial Requirements	50
4.2	Schedules	50
5	Closure Materials	53
5.1	Project Evaluation.....	53
5.2	Commercialization Possibility	55
5.3	Recommendations for Additional Work.....	55
5.4	Lessons Learned.....	56
5.4.1	What Went Well	56
5.4.2	What Did Not Go Well	56
5.4.3	What Technical Knowledge Was Gained	56
5.4.4	What Non-Technical Knowledge Was Gained.....	56
5.4.5	What The Team Would Do Differently If The Team Had To Do It Over Again	56
5.5	Risk and Risk Management	56
5.5.1	Anticipated potential risks and planned management	57
5.5.2	Anticipated risks encountered and success in management	57
5.5.3	Unanticipated risks encountered, attempts to manage and success	57

5.5.4	Resultant Changes in Risk Management Made Because of Encountered Unanticipated Risks	57
5.6	Project Team Information	57
5.7	Client Information.....	58
5.8	Advisor Information.....	58
5.9	Closing Summary.....	58
Appendix A.	Sample Test Data	60
Appendix B.	Sample GUI Setting File.....	61
Appendix C.	Sample Simulink Block Diagram	64
Appendix D.	Detailed Flowchart.....	65

List of Tables

Table 1.1: Initial Assumptions and Justifications.....	3
Table 1.2: Initial Limitations and Justifications.	4
Table 2.1 Product Requirements and Features	5
Table 3.1 Sample Test Data Format	30
Table 3.2 Data Structures and their Fields.....	33
Table 3.3 Product Requirements and Features	38
Table 4.1 Original Labor Estimates	46
Table 4.2 Actual Labor Usage	48
Table 4.3 Financial Costs Associated with Resources (Original Estimate).....	50
Table 4.4 Financial Costs Associated with Resources (Actual Costs)	50
Table 5.1 Project Task Evaluation	53
Table 5.2 Milestone Evaluation	55

List of Figures

Figure 3.1 Simplified Flow of Information.....	28
Figure 3.2 Simplified Flow Chart	29
Figure 3.3 Detailed Flow of Information.....	31
Figure 3.4 Main Window	35
Figure 3.5 Home Window.....	36
Figure 3.6 Wait bar	38
Figure 3.7 Simulation Error Message Box	40
Figure 3.8 Batch Simulate Window.....	41
Figure 3.9 Open Window.....	42
Figure 3.10 Save As Window	43
Figure 4.1 GANTT Chart Legend.....	51
Figure 4.2 Fall 2006 Schedule	51
Figure 4.3 Spring 2007 Schedule.....	51
Figure 4.4 Deliverables Schedule	52

List of Definitions

- Block diagram – A visual method commonly used in engineering to depict the relationship between inputs and outputs of a system.
- Exciter – System that generates the rotor currents needed to produce electric flux.
- Genrou – Round rotor General Electric generator model.
- Gensal – Salient pole General Electric generator model.
- GUIDE – Graphical User Interface Development Environment – A Matlab toolbox that supplies a graphical and interactive method for the creation of graphical user interfaces.
- Initial Conditions – Values that determine the simulation's starting point.
- Matlab – A software package developed by Mathworks that is commonly used for engineering computation.
- Matlab Workspace – The location where all variables are stored.
- Physical test results – These are the results of actual generator performance as physical measured in the field.
- PSLF – (Positive Sequence Load Flow) – A software tool manufactured by General Electric that is used by power systems engineers to analyze the performance and security of large interconnected power systems.
- PSLF test results – These are the results of control simulations run using PSLF.
- Semi-real time – Result updates appear after the user selects the run command.
- Setting File – GUI reads setting files to customize it's appearance.
- Simulation Results – Output data from the Simulink based generator models
- Simulink – A Matlab toolbox that allows users to build and analyze block diagrams.
- Test results – Data collected in the field that is compared to simulation results.

Simulink and Matlab are copyrighted by The Mathworks Inc.

Windows is copyrighted by Microsoft Corporation

PSLF is copyrighted by General Electric Company

1 Introduction

This section defines the purpose and major objectives of the project, including a project description, discussion of operating environment, intended users and uses, assumptions and limitations and expected end product and deliverables.

1.1 Executive Summary

In order to analyze the security and stability of a power network, engineers use software such as General Electric's PSLF to test their generators under various conditions. In order for these simulations to be accurate, detailed models of synchronous generators are needed. These models have a great number of parameters that are determined by testing the machines under known conditions in the field and varying simulated parameters based on the engineer's intuition and experience until the model produces the same results as the field test. Although the field tests only take a few hours, the process of determining generator model parameters can take an experienced engineer between three and five days. Software was needed to help engineers determine these parameters in less than a day.

1.1.1 Project Activities

The project was divided up into 4 main phases. The first phase consisted of research on generators and collecting information from the client. The second phase consisted of planning the design process. The third and longest phase was an iterative process of prototyping and feedback from the client. Followed by fine tuning, bug testing and documentation in the fourth and final phase.

1.1.2 Final Results

The project team created a software package that met or exceeded the expectations of all required and supplemental specifications. It allows engineers to determine the parameter settings on their generators much more efficiently. The team feels that the project is a tremendous success and will save the company time and money.

1.1.3 Recommendations for Additional Work

There are several features that were outside the scope of this project. A wizard for adding new generator models is one such feature. Another feature that could be created that would take significant work is an algorithmic parameter analyzer that would somewhat automate the process of machine parameter determination.

1.2 Acknowledgements

The project team would like to thank Doug Welsh, Dan Leonard and Juan Sanchez of General Electric and Dr. Chen-Ching Liu of Iowa State University for the assistance and technical expertise that they have provided.

1.3 Problem Statement

The general project statement and general project approach explain the problem and the approach that the team used for the solution.

1.3.1 General Problem Statement

Accurate models of synchronous generators are essential to assess the stability of a power system. The current process used by General Electric to determine machine model parameters consists of running tests under known conditions in the field and recording extensive amounts of data related to machine performance. After returning to the office, General Electric engineers estimate model parameters using the engineer's own intuition and experience, load these into PSLF, and run simulations of the field tests. The simulation results are then compared to the results of the actual field tests. The engineer iteratively matches the test data to the simulation data until an accurate parameter setting for the generator is determined. Although the data collection for field tests only take a few hours, the process of determining accurate model parameters takes an experienced engineer between three and five days with currently used PSLF software.

The client desires a well documented, interactive tool designed using Simulink and Matlab that will reduce the time required to determine generator parameters to less than 4 hours.

1.3.2 General Solution Approach

The project team met with the client and faculty advisor to develop a more complete understanding of the project and create a prototype block diagram and graphical user interface. Various approaches to both the design of the GUI and machine models were considered. The end design uses a Matlab based GUI that allows the user to select a generator type, set generator parameters and view output waveforms. Simulink models of each generator take the parameters from the GUI and calculate the output waveforms.

Other approaches that were considered included using the Java programming language to create the GUI. Java is excellent for creating GUIs but does not interface with the Matlab environment very well. Although the Matlab language is not as well set up to design complex GUIs it interfaces with the Simulink models more efficiently. In the end, it was decided to use Matlab because the additional features available in the Java interface did not provide any benefits in this context.

An end user manual, test plan, and this final report are provided to ensure that the client fully understands how to use the product, how it was tested for robustness, and add additional machine models to the interface in the future.

1.4 Operating Environment

The product was designed using Simulink and Matlab2006a/Matlab2006b, it will be run on Windows XP based computers with monitor resolution greater than 1024x768. Matlab and Simulink can be run on other operating systems and screen resolutions, but has only been tested in this environment.

1.5 Intended Users and Uses

This section defines the intended users and uses of the project team's solution.

1.5.1 Intended User(s)

This software will be used by exclusively by employees of General Electric. It is assumed that they are experienced engineers who are familiar with Simulink and Matlab, generator stability, and machine testing.

1.5.2 Intended Use(s)

This software is used to aid engineers in quickly determining model parameters for synchronous machines.

1.6 Assumptions and Limitations

This section defines the initial assumptions and limitations that the group worked from to solve the problem.

1.6.1 Initial Assumptions

Table 1.1 lists the initial assumptions that the group worked from to find and implement the solution to the problem.

Table 1.1: Initial Assumptions and Justifications.

Assumptions	Justification
The project team will be provided with models of all relevant machines to be included before project completion.	Implementation
The project team will have access to the results of relevant PSLF simulations for results comparison.	Testing
Project team shall not share information supplied by client or design results with outside individuals or groups.	Confidentiality
Client's version of Matlab will be compatible with project team's.	Design
The program interact with the user in a graphical way.	Design

1.6.2 Initial Limitations

Table 1.2 lists the initial limitations with which the group will comply.

Table 1.2: Initial Limitations and Justifications.

Limitations	Justification
The software will be created using Matlab and Simulink.	Client
Users will be able to quickly adjust all machine model parameters using up and down arrows as well as text boxes.	Client
Time required to obtain accurate model parameters will not exceed four hours.	Client
Completed project will include software, user manual, test plan/results and this final report.	Design
The output graphs contained in the GUI will update by buttons contained within the GUI itself.	Design

1.7 Expected End Product and Other Deliverables

This section lists all items that are supplied to the client as part of the end product.

1.7.1 Simulink Block Diagram Files

Fully functional Simulink files of all machine types for which the group has block diagrams for, are included in the software package. These files conform to the limitations described in Table 1.2.

1.7.2 Setting Files

Setting files for each generator type will be included in the software package. These setting files tailor the layout of the graphical user interface for each model type.

1.7.3 Graphical User Interface

A graphical user interface is provided in the form of a Matlab “m-file.” It interacts with the included Simulink models and setting files. The source code will also be provided.

1.7.4 Project Documentation

The project team will deliver electronic and bound copies of the final reports and end user manual to ensure that the client understands the software design and its proper and most effective use as well as the design process.

2 Approach and Results

This section outlines the design team's solution to the client's problem, steps that were taken for the design to be successful, as well as the functional requirements for the final product.

2.1 Approach

The project is deemed successful if it fulfills the following requirements.

2.1.1 Functional Requirements

The required and supplemental functionalities are listed in Table 2.1.

Table 2.1 Product Requirements and Features

Feature	Required or Supplemental	Intended Functionality
Edit Box	Required	Provide the user with a means to quickly enter a specific value for a parameter.
Slider Bar	Required	Provide the user with an easy method to vary parameters. Each slider bar shall have a default maximum and minimum value. When the slider bar position is changed, the value in the edit box shall update. If a value is entered into the edit box that is outside of this range, the corresponding maximum or minimum should reset to the value in the edit box.
Interface with Simulink	Required	The user shall be able to execute Simulink models from the graphical user interface.
Graph Simulation Data	Required	Simulation data shall be plotted on figures in the graphical user interface.
Import Data from Excel and Display on Graphs	Required	The user shall be able to import time domain data stored in Microsoft Excel spreadsheets in a format specified by the project team and plot them on the same graphs as the simulation data.
Undo/Redo Actions	Required	The user shall be able to undo actions upto to previous simulation.
Save Current Session	Required	The user shall be able to save the position of each slider bar.
Load Previous Session	Required	The user shall be able to load the position of each slider bar from a previous session.
Zoom In and Out on Graphs	Required	The user shall be able to zoom in and out on all graphs.

Table 2.1 Continued.

Feature	Required or Supplemental	Intended Functionality
Handle Simulation of Different Tests	Required	The graphical user interface shall be able to handle models capable of simulating different events
Handle Multiple Models	Required	The graphical user interface shall be capable of handling a library of any number of models. The project team shall enter at least 5 models into this library.
Pan Graphs	Supplemental	The user should be able to pan graphs.
Save Graphs as Matlab Figures	Supplemental	The user should be able to save graphs as Matlab figures.
Export Simulation Data to Excel Spreadsheet	Supplemental	The user should be able to save simulation data as an Excel spreadsheet in a format that is the same as the imported data.
Batch Simulation	Supplemental	The user should be able to choose a single parameter, specify a minimum and maximum value, and any number of simulations and have the program iterate that many times between those two points and graph the simulation data.
Hold Previous Zoom Level	Supplemental	When test data has been imported, the Graph's zoom level should not change after the simulate button is pressed.
Graph Previous Simulation Data	Supplemental	The previous simulation data should appear as a light gray line to allow the user to judge progress in matching the test data.
Color Code for Edit Box	Supplemental	Color edit boxes to help user to visually group different types of parameters belonging to different subsystems.

2.1.2 Design Constraints

The constraints outlined below must be considered:

- The program must run completely within the Matlab R2006a/R2006b program environment, including the machine model running in the Simulink environment.
- If the parameters are varied, the output displays must be able to update in semi-real time. The results must appear within a few moments.

- The complexity of the block diagram and number of parameters is generator specific. The GUI for each generator will be tailored to its specific parameters and outputs, while maintaining standardization.

2.1.3 Technical Approach Considerations

The technical aspect of the project was approached in the following way:

- Full generator models were provided by the client for the team to integrate into the software package. It was determined that the team did not have the technical expertise to create models of General Electric's generators from scratch in the amount of time provided.
- Although there is some supplementary Matlab text based code, the generator models were created using Simulink in graphical block diagram form so that the structure is apparent.
- The graphical user interface window contains everything that the user needs to easily vary the parameters and analyze the outputs of the machine. It was determined that having all of the parameter values and outputs plots in one window would provide a faster and simpler interface for the engineers. Having multiple windows was considered, but determined to be less efficient.
- The software package runs entirely from the Matlab program. It was determined that although other languages such as Java could offer more flexibility in the design, the Matlab graphical environment would perform the required functions as well as interacting with the Simulink files in a more efficient way.
- The Matlab workspace is the holding place for all variables and other data passed between the graphical user interface and Simulink models. A direct link was considered between the GUI and models. This approach was not chosen because the intermediary space is easily accessible by both, which allowed for more efficient software coding. Also, in the event of a simulation or GUI error, the current data will not be lost.
- Output graphs are of the same form as those generated by testing and are displayed in the GUI in such a way that they can be easily compared by the user. It was determined that the graphical user interface should be able to handle different time step values for the simulation results to match the precision of test results from the field.
- The parameter slider bars and entry boxes are color coded to represent their function in the machine and separate them visually to the user. Testing revealed that separating the parameters boxes with white space and headings took too much space away from the output plots when used in a low monitor resolution. The client specified that the large size of the plots was more important for accuracy.
- Setting files have been created for each machine detailing which parameters will be in use for all available tests. Parameters edit boxes that are in use for a

machine, but not for the selected test are disabled. It was determined the GUI should only display the parameters edit boxes that apply to that machine in order to save space for the output plots. It was also determined that removing and replacing parameter edit boxes for each individual test within a machine would slow down the user experience too much.

2.1.4 Implementation Process Description

After developing a robust and usable prototype, the primary implementation activities for the project consisted of obtaining models, settings, and parameters for all of the different generators that were going to be used in the program. A significant hurdle was overcome early in the implementation process through the development of setting files and parameter files that interface with the main GUI. This allowed the team to rapidly develop the necessary configuration settings for each model without the need to change code to the top level program.

Additionally, the close relationship between the team and the client allowed a continuing process of product improvement throughout the implementation process. Several key features not originally discussed were added to the program during the implementation that improved functionality and usability. Some of these functions were batch simulation processing, data exportation and undo and redo functions.

2.1.5 End Product Testing

The overall testing process will be broken down into two parts:

- Model testing
- Functional testing

Model testing outlines the procedures to validate the correctness and accuracy of the Simulink models. Functional testing outlines the procedures to verify that all functional requirements and features have been implemented correctly.

2.1.5.1 Model Testing

The purpose of model testing is to verify that:

- The Simulink models provide accurate results.
- The Simulink models are able to provide results over the entire normal range.

2.1.5.1.1 Scope and Approach

Model testing will include all testing activities necessary to verify that the models produce accurate and reliable results. Each model included in the library by March 21, 2007 was tested individually.

2.1.5.1.1.1 Models Included in Model Testing

The following models were tested:

- Unexcited genrou model.
- Genrou model with exst4b exciter.
- Genrou with rexs exciter.
- Unexcited gensal model.
- Gensal with exst1 exciter.
- Gensal with exst1 exciter and HVGOV governor.

2.1.5.1.1.2 Items Included in Model Testing

The following items were included in model testing:

- Verification of default edit box values.
- Simulation with all parameters set to default values.
- Verification of minimum edit box values.
- Verification of maximum edit box values.
- Verification that varying parameters produces different simulation results.
- Comparison of simulation data with PSLF simulation results.

2.1.5.1.1.3 Items Excluded from Model Testing

The following items have been excluded from model testing:

- Testing to determine what values of certain parameters give numerical errors.
- Testing to determine if certain combinations of parameters give numerical errors.

2.1.5.1.2 Test Cases

This section summarizes the tests that were performed for each model. All tests not related to verifying edit box and slider bar values were performed on Matlab versions R2006a and R2006b. The test procedure outlined in Section 2.3.6 describe the process that the Iowa State team feels would verify that the models produce accurate numerical results. Because the Iowa State team does not have access to the software necessary to complete this kind of testing, it will be completed by the team at General Electric. The exact procedure used was left to their judgment.

The test case defines what was tested. The reason for testing this is described in the purpose. Procedure outlines how the test were performed. The expected result explains what result was considered completely successful. The actual result describes what happened when the test was executed and what modifications were necessary to achieve the expected result. If the expected result was not achieved and no attempt to remedy the situation was made, a justification is given.

2.1.5.1.2.1 Verification of Default Edit Box Values

Test Case: Verify default edit box values.

Purpose: To verify that the default edit box values match the data sent by General Electric.

Procedure: Open each model from the opening window and compare the default edit box values to the data sent by General Electric.

Expected Result: All default edit box values for all models should match the data sent by General Electric.

Actual Result:

R2006a: All included models displayed the correct default value

R2006b: All included models displayed the correct default value

Action Required: none

Completed: yes

2.1.5.1.2.2 Simulation with All Parameters Set to Default Values

Test Case: Simulate all models with the parameters set at their default values.

Purpose: To verify that all models will simulate with the parameters set at the default values.

Procedure: Open each model from the opening window and use the slider bar to set all values to the maximum. Compare the edit box values to the data sent by General Electric.

Expected Result: All maximum edit box values for all models should match the data sent by General Electric.

Actual Result:

R2006a: Same error as R2006b, except for some computers there is an error "Reference to non-existent field 'undo_toolbar_button'." When running a simulation.

R2006b: Each model simulates with parameters set to default. However if you open a new model without first clearing the workspace there is a simulation error.

Action Required: Insert command to clear workspace variables on main_window.m startup. Investigate fatal error on R2006a if time permits, however error does not occur on majority of computers. After further inspection it, it appears that this only occurs on corrupt Matlab installations and is not a problem with the GUI.

Completed: Yes

2.1.5.1.2.3 Verification of Minimum Edit Box Values

Test Case: Verify minimum edit box values.

Purpose: To verify that the default minimum edit box values match the data sent by General Electric.

Procedure: Open each model from the opening window and use the slider bar to set all values to the minimum. Compare the edit box values to the data sent by General Electric.

Expected Result: All minimum edit box values for all models should match the data sent by General Electric.

Actual Result:

R2006a: All included models displayed the correct default value

R2006b: All included models displayed the correct default value

Action Required: none

Completed: yes

2.1.5.1.2.4 Verification of Maximum Edit Box Values

Test Case: Verify maximum edit box values.

Purpose: To verify that the default maximum edit box values match the data sent by General Electric.

Procedure: Open each model from the opening window and use the slider bar to set all values to the maximum. Compare the edit box values to the data sent by General Electric.

Expected Result: All maximum edit box values for all models should match the data sent by General Electric.

Actual Result:

R2006a: All included models displayed the correct default value

R2006b: All included models displayed the correct default value

Action Required: none

Completed: yes

2.1.5.1.2.5 Verification that Changing Parameters Produces Different Simulation Results

Test Case: Verify that changing the edit box produces different simulation results.

Purpose: To verify that changing the edit box produces different simulation results.

Procedure: Open each model from the opening window and press simulate. One by one use the slider bar to set a parameter values to a value that is very different from its default value. Press the simulate button and use the previous simulation trace to verify that at least one of the curves changed.

Expected Result: All slider bars corresponding to parameters other than those belonging to saturation points will produce different results.

Actual Result:

R2006a: Same as R2006b

R2006b:

Voltage Step Test:

Rcomp and Xcomp on Genrou model has no effect on output.

Vrmax on Genrou Exst4b has no effect on output. Depending on other parameter settings, simulation when Lpd is .1 sometimes results in simulation error.

Tr, Vrmax, Kf, Tf, Tf1, Tf2, Fbf, Kii, Vfmax, Kh, E1, Se1, E2, Se2 on Genrou Rxs has no effect on output.

Rcomp on Gensal has no effect on output.

Vimax on Gensal ext1 has no effect on output . Tb=0 simulation error

Tr, Vrmax, Tf, Llr, Klr, Velm, Gmax, Gmin on Gensal exst1 + hygrov has no effect on output.

Action Required: Simulation error cases are result of internal mathematics, no changes necessary. Verify with GE which parameters should be disabled for certain tests and include this information in setting file. After discussion with GE, it appears that they do not want these parameters disabled.

Completed: Yes

2.1.5.1.2.6 Comparison of Simulation Data with PSLF Simulation Results

Test Case: Simulate all models with the parameters set at their default values and compare to PSLF data.

Purpose: To verify that all models produce correct numerical results at their default values.

Procedure: Use PSLF to prepare test data corresponding to the default case. Export this data to a Microsoft Excel Spreadsheet whose format matches the one described in the detailed design. Open each model from the opening window import the test data. Press the simulate button. Compare the simulation results with the PSLF results.

Expected Result: All plots will match within a reasonable tolerance.

Actual Result: To be completed by GE.

Test Case: Simulate all models with the parameters set at a value other than its default and compare to PSLF data.

Purpose: To verify that all models produce correct numerical results at a value other than their default value.

Procedure: Use PSLF to prepare test data corresponding to a case in which the all parameters would not be set at their defaults. Export this data to a Microsoft Excel Spreadsheet whose format matches the one described in the detailed design. Open each model from the opening window import the test data. Change each edit box to the correct value. Press the simulate button. Compare the simulation results with the PSLF results.

Expected Result: All plots will match within a reasonable tolerance.

Actual Result: To be completed by GE.

2.1.5.2 Functional Testing

The objective of functional testing is to verify that:

- The software met the functional requirements specified in the project description and design report.
- All features required for each functional requirement are implemented correctly.
- All features function properly in the possible operating environments.

2.1.5.2.1 Scope and Approach

The scope of functional testing is to verify that the functional requirements and sets of features perform as intended.

For each functional requirement, there will be a set of test cases and corresponding expected results for each feature associated with that requirement. The actual test results are then compared with the expected results and documented in 2.1.5.2.2 for future reference.

2.1.5.2.1.1 Items Included in Functional Testing

The following is a list of functional requirements included in functional testing:

- Start/exit the program from Matlab
- Select a machine model to simulate from main window
- Import test data and export simulation data.
- Save and open simulation data.
- Edit parameter values using the edit boxes and slider bars.
- Undo/redo changes using undo and redo buttons.
- Plot simulation results.
- Zoom and pan plots.

- Overall appearance for different screen sizes.
- Switching between different machine tests.
- Batch simulations.
- Save output graphs as Matlab figures.
- Graph previous simulation data.
- Time required to determine the parameters should be less than four hours.
(This will be completed by the team at General Electric.)

2.1.5.2.1.2 Items Excluded from Functional Testing

Here is the list of requirements excluded in functional testing:

- Non-functional requirements such as ease-of-use.
- Setting files are tested under model testing and are assumed to be properly configured.

2.1.5.2.2 Test Cases

Here is the list of test cases and expected results for each functional requirement listed in 2.1.5.2.1.1. The test case defines what will be tested. The reason for testing this is described in the purpose. Procedure outlines how the test will be performed. The expected result explains what result will be considered completely successful. The actual result describes what happened when the test was executed and what modifications were necessary to achieve the expected result. If the expected result was not achieved and no attempt to remedy the situation was made, a justification should be given.

2.1.5.2.2.1 Start and Exit Program From Matlab

Test Case: Start program from Matlab.

Purpose: To verify user can start the program correctly by following the instructions outlined in the user manual.

Procedure: Have testers without knowledge of the software follow the procedure outlined in the user manual to start the program inside Matlab.

Expected Result: Tester should be able to start the program within Matlab and opening window should appear correctly.

R2006a: Same as R2006b

R2006b: Issues found, see below.

Action Required: Highlight/circle current directory in Fig 2.1 of user manual

Completed: Yes

Test Case: Exit opening window.

Purpose: To verify user can exit opening window using the close button and that the opening window closes properly.

Procedure: Close opening window using the close button in the upper right hand corner.

Expected Result: Opening window closes properly.

Actual Result:

R2006a: Closed properly.

R2006b: Closed properly.

Action Required: none

Completed: Yes

2.1.5.2.2.2 Select Machine Model to Simulate From Opening Window

Test Case: Select machine model from opening window.

Purpose: To verify the function of selecting machine model from opening window has been correctly implemented.

Procedure: Select each machine model in list box and click open button in opening window.

Expected Result: Home screen corresponding to the selected model should appear correctly and opening window should close after the home screen appears.

Actual Result:

R2006a: Opened properly.

R2006b: Opened properly.

Action Required: none

Completed: Yes

2.1.5.2.2.3 Import and Export Test and Simulation Data

Test Case: Import test data from a Microsoft Excel file.

Purpose: To verify that the program can import test data from a properly formatted Excel spreadsheet for each machine test.

Procedure: For each machine test, click the open button in the upper right corner of home window, browse for the test data and click ok. Test data should be in Excel format specified in the detailed design section.

Expected Result: Test data is plotted correctly in the output panel of home window for each machine test.

Actual Result:

R2006a: Opened Properly.

R2006b: Opened properly.

Action Required: none

Completed: Yes

Test Case: Import invalid test data.

Purpose: To verify home window still functions properly after attempting to import improperly formatted test data.

Procedure: Prepare test data formatted in a format other than the format described in the detailed design and attempt to import it.

Expected Result: Exception message is displayed and the program still functions properly.

Actual Result:

R2006a: Same as R2006b

R2006b: When invalid numbers were put into time column or plot data column, no errors occurred, just gaps in plot. When columns were in wrong location, a detailed error message was given.

Action Required: none

Completed: Yes

Test Case: Export simulation data after simulation.

Purpose: To verify export simulation data feature has been correctly implemented.

Procedure: Click save button and verify that exported simulation data matches the simulation results.

Expected Result: Exported data matches simulation results displayed on the home screen.

Actual Result:

R2006a: Same as R2006b

R2006b: .Jpg saves correctly however legend is missing.

Action Required: Add legend to .jpg captures.

Completed: Yes

Test Case: Export simulation data before simulation.

Purpose: To verify program still functions properly after attempting to export simulation data when none exists.

Procedure: Click export button before running any simulations when home window starts.

Expected Result: Exception message appears and program still functions properly.

Actual Result:

R2006a: Same as R2006b

R2006b: .Jpg saves empty plots (non-issue). .xls does not save, no error message given.

Action Required: Add exception to display message notifying user that .xls was not saved.

Completed: Yes

2.1.5.2.2.4 Save and Open Parameter Data

Test Case: Save parameter data.

Purpose: To verify that the save parameter data function has been correctly implemented.

Procedure: Use the save button to save the data as a Matlab data structure. Use Matlab to view the data structure. Check to see if the data in the structure is correct.

Expected Result: Data will be saved correctly.

Actual Result:

R2006a: Same as R2006b

R2006b: Parameter data saves correctly.

Action Required: none

Completed: Yes

Test Case: Open previously saved parameter data.

Purpose: To that the user can open saved parameter data and that the edit boxes and slider bars are set to the correct position.

Procedure: Use the open button to import previously saved parameter data. Check to see that the edit boxes now display the correct values. Check that the slider bars are in the correct position.

Expected Result: User can import previously saved data and that data will be displayed properly on the home window.

Actual Result:

R2006a: Same as R2006b

R2006b: Parameter data opens correctly.

Action Required: none

Completed: Yes

Test Case: Open previously saved parameter data from a different machine.

Purpose: To verify that the program works properly after the user attempts to import parameter data from a machine with the incorrect number of parameters.

Procedure: Use the open button to import previously saved parameter data. Check to see if the exception is handled, the user warned, and that the program continues to run properly.

Expected Result: The exception will be handled, the user will be warned, and the program will continue to run properly.

Actual Result:

R2006a: Same as R2006b

R2006b: Non-detailer message pops up.

Action Required: Reword error message to say “Error loading parameter settings or parameter settings for wrong machine.”

Completed: Yes

2.1.5.2.2.5 Edit Parameters Values Using Edit Boxes and Slider Bars

Test Case: Check default values for all edit boxes

Purpose: To verify default values match data in setting file.

Procedure: Open the home window and check that the default values listed for each parameter are the same as those in the setting file.

Expected Result: Default values in edit box match those in the setting file.

Actual Result:

R2006a: Same as R2006b

R2006b: Setting files match

Action Required: none

Completed: Yes

Test Case: Enter valid value in edit box

Purpose: To verify that the edit boxes have been properly implemented.

Procedure: Enter various valid numerical value into the edit box, values entered should include values inside and outside the minimum and maximum range specified in the setting file. Place a breakpoint inside code to verify that the minimum and maximum values are properly reset when value outside the minimum and maximum range is entered.

Expected Result: Slider bar should move accordingly to the entered numbers, the minimum and maximum values should reset properly when a value outside of the default range is entered.

Actual Result:

R2006a: Edit boxes act correctly.

R2006b: Edit boxes act correctly.

Action Required: none

Completed: Yes

Test Case: Enter invalid value in edit box

Purpose: To verify the program gives proper warning message and continues to function properly after an invalid (non numeric) value is entered in the edit box

Procedure: Enter various non numeric values into the edit box.

Expected Result: Warning message of invalid value should appear and the program should continue to function properly.

Actual Result:

R2006a: Same as R2006b

R2006b: Entering invalid value resets the edit box to previous value.

Action Required: This is acceptable, none.

Completed: Yes

Test Case: Change parameter value using sliderbar

Purpose: To verify sliderbar is implemented properly

Procedure: Move slider between min/max and verify value inside edit box is being updated accordingly.

Expected Result: Value inside edit box updates properly according to the sliderbar position.

Expected Result: Value inside edit box updates properly according to the sliderbar position.

Actual Result:

R2006a: Sliders act correctly.

R2006b: Sliders act correctly.

Action Required: none

Completed: Yes

2.1.5.2.2.6 Undo and Redo Changes Using Undo and Redo Button

Test Case: Undo changes using undo button.

Purpose: To verify that undo button has been properly implemented.

Procedure: Press the simulate button, change 20 parameters noting previous and current values, press the simulate button, then click the undo button 20 times. Verify that the values are reset to the correct value.

Expected Result: Parameter values change back to previous values each time undo button is pressed.

Actual Result: Actual Result:

R2006a: Undo button performs correctly.

R2006b: Undo button performs correctly.

Action Required: Identify undo/redo disabling code and check for uniformity.

Completed: Yes

Test Case: Redo changes using redo button.

Purpose: To verify redo button is implemented properly

Procedure: Press the simulate button, change 20 parameters noting previous and current values, press the simulate button, then click the undo button 20 times. Verify that the values are reset to the correct value, then click the redo button 20 times, verify that the values are set to the correct values.

Expected Result: Parameter values change back to previous values each time redo button is pressed.

Actual Result:

R2006a: Redo button performs correctly.

R2006b: Redo button performs correctly.

Action Required: Identify undo/redo disabling code and check for uniformity.

Completed: Yes

2.1.5.2.2.7 Zoom and Pan Plots

Test Case: Zoom in on graphs without test data.

Purpose: To verify zoom in feature is implemented properly.

Procedure: Run simulation without test data loaded, zoom in or out on the graphs by pressing the zoom in button and using the left mouse click. Zoom out using the shift left click, simulate one more time after zooming in or out to ensure graphs reset properly.

Expected Result: Graphs zoom in and out properly, graphs reset properly after each simulation.

Actual Result:

R2006a: Works correctly.

R2006b: Works correctly.

Action Required: none

Completed: Yes

Test Case: Zoom in on graphs with test data.

Purpose: To verify zoom in feature is implemented properly.

Procedure: Run simulation with test data loaded, zoom in on the graphs by pressing the zoom in button and using the left mouse click. Zoom out using the shift left click, simulate one more time after zooming in or out to ensure graphs reset properly.

Expected Result: Graphs zoom in and out properly, graphs do not reset after each simulation.

Actual Result:

R2006a: Works correctly.

R2006b: Works correctly.

Action Required: none

Completed: Yes

Test Case: Zoom out on graphs without test data.

Purpose: To verify zoom out feature is implemented properly.

Procedure: Run simulation with out test data loaded, zoom out on the graphs by pressing the zoom out button and using the left mouse click. Zoom in on graphs

using the shift key plus left click, simulate one more time after zooming in and out to ensure that the graphs hold the current zoom level.

Expected Result: Graphs zoom in and out properly, graphs reset after each simulation.

Actual Result:

R2006a: Works correctly.

R2006b: Works correctly.

Action Required: none

Completed: Yes

Test Case: Pan plots without test data.

Purpose: To verify that the pan feature is properly implemented.

Procedure: Run simulation without test data loaded, pan graphs in different directions after zooming in, simulate afterward to ensure that graphs reset.

Expected Result: Graphs pan properly and reset after simulation.

Actual Result:

R2006a: Works correctly.

R2006b: Works correctly.

Action Required: none

Completed: Yes

Test Case: Pan plots with test data.

Purpose: To verify that the pan feature is properly implemented.

Procedure: Run simulation with test data loaded, pan graphs in different directions after zooming in, simulate afterward to ensure that graphs do not reset.

Expected Result: Graphs pan properly and hold current position after simulation.

Actual Result:

R2006a: Works correctly.

R2006b: Works correctly.

Action Required: none

Completed: Yes

2.1.5.2.2.8 Overall Appearance for Different Screen Sizes

Test Case: Slider bar and edit box appearance at large and small screen size.

Purpose: To verify slider bars and edit boxes appear and resize correctly.

Procedure: Start home window with various numbers of parameters in setting file and resize window by clicking and dragging the bottom right corner.

Expected Result: All slider bars and buttons show up completely and are clickable, text is readable in edit box and is shown to at least 4 significant digits. Labels are also readable.

Actual Result:

R2006a: Same as R2006b

R2006b: Appearance is sufficient at minimum of 1024x768 pixels.

Action Required: none

Completed: Yes

Test Case: Graph appearance at large and small screen size.

Purpose: Verify that graphs appear correctly.

Procedure: Run program with setting files that have different numbers of output graphs to determine the maximum feasible number of output plot boxes.

Expected Result: All plots, labels and axes appear correctly up to a maximum value.

Actual Result:

R2006a: Same as R2006b

R2006b: Appearance is sufficient at minimum of 1024x768 pixels. Smaller windows still function, however there are some readability problems.

Action Required: none

Completed: Yes

2.1.5.2.2.9 Switching Between Different Machine Tests

Test Case: Select machine test from drop box

Purpose: Verify that correct edit box are disabled for each machine test, and that a different event is simulated for each position.

Procedure: Select each machine test, simulate, verify that the correct boxes are editable, and that the simulation results are logical.

Expected Result: Edit box are disabled properly, a different test is simulated, and the simulation results make sense for each test selected.

Actual Result:

R2006a: Functions correctly, see note.

R2006b: Functions correctly, see note.

Action Required: (note) Disabled edit boxes need to be applied to exciters and governor parameters as discussed in “Verification that Changing Parameters Produces Different Simulation Results” section.

Completed: Per a discussion with the client, the edit boxes will not be disabled for exciter and governor parameters.

2.1.5.2.2.10 Batch Simulations

Test Case: Batch simulation with the number of simulations set to a number less than 1.

Purpose: To verify that batch simulation has been properly implemented.

Procedure: Click batch simulation button in lower left corner of home window, choose a single parameter enter 0 into the simulations box and click OK.

Expected Result: One simulation is run and the graphs appear properly with the correct legend.

Actual Result:

R2006a: Same as R2006b

R2006b: Any number of simulations less and or equal to one is reset to two simulations.

Action Required: none

Completed: Yes

Test Case: Batch simulation with the number of simulations set to a non integer number.

Purpose: To verify that batch simulation has been properly implemented.

Procedure: Click batch simulation button in lower left corner of home window, choose a single parameter enter 3.25 into the simulations box and click OK.

Expected Result: Three simulations are run and the graphs appear properly with the correct legend.

Expected Result: Three simulations are run and the graphs appear properly with the correct legend.

Actual Result:

R2006a: Same as R2006b

R2006b: Any number of simulations that are a non integer are rounded to the nearest integer.

Test Case: Batch simulation with the number of simulations set to a positive integer number.

Purpose: To verify that batch simulation has been properly implemented.

Procedure: Toggle a breakpoint in the home screen code. Click batch simulation button in lower left corner of home window, choose a single parameter enter 4 into the simulations box and click OK. Check to see that the correct parameter is being varied and that the results are plotted correctly with the correct legend.

Expected Result: Four simulations are run and the graphs appear properly with the correct legend.

Expected Result: Four simulations are run and the graphs appear properly with the correct legend.

Actual Result:

R2006a: Preforms correctly.

R2006b: Preforms correctly.

Action Required: none

Completed: Yes

2.1.5.2.2.11 Save Output Plots as .jpg Files

Test Case: Save output plots as JPEG images.

Purpose: To verify that the save output plot feature has been correctly implement.

Procedure: Run a simulation, click the save button and save the output plots as test.jpg. Look in the directory to see if the correct number of figure files are there and that the filenames are correct. Use windows to open the figures and compare them to the output plots.

Expected Result: The correct number of figures will be created with the filenames as described in the detailed design and the figures will match the output graphs.

Actual Result:

R2006a: Same as R2006b

R2006b: .Jpg saves correctly however legend is missing.

Action Required: Add legend to .jpg captures.

Completed: Yes

2.1.5.2.2.12 Graph Previous Simulation Data

Test Case: Graph previous simulation data.

Purpose: To verify that the previous simulation data is graphed correctly.

Procedure: Run a simulation, change the event time from the default to 5 seconds, press the simulate button. Check to see that the previous simulation data is plotted with the correct event time as a light gray line and that the current data appears as a colored line with the same numerical results at a time delay. Verify that the correct legend appears.

Expected Result: The previous simulation results should appear as a light gray line with the default event time. The current simulation data should appear as a colored line with the same magnitudes as the previous simulation, but with a time delay. The legend should be correct.

Actual Result:

R2006a: Works correctly.

R2006b: Works correctly.

Action Required: none

Completed: Yes

2.1.5.2.2.13 Time Required to Determine Machine Parameters

Test Case: Test the speed with which a set of parameters can be found using software package.

Purpose: To verify that the time requirement is met.

Procedure: Have a member of the General Electric team determine machine parameters using test data from an unknown generation facility. Time the engineer.

Expected Result: Parameters will be determined in less than 4 hours.

Actual Result: Preliminary results done by GE indicate expected result is on target, more testing will be needed to confirm.

2.2 Project End Result

The final outcome of the project was a complete success. The client was able to accurately match the programs simulation values to a more cumbersome, but very accurate simulation program that has been in place with the company. Additionally the client was able to take fresh data from the field and make use of this program to analyze the parameters of the generator. On the team's side, everyone felt as though the experience was very helpful because of the significant Matlab and Simulink knowledge learned will surely help in the future.

3 Detailed Design

This section is intended to provide the reader with an in-depth background in the program's structure and design. Selected components are referenced in the appendices of this document.

3.1 Design Overview

The software was developed in the Matlab programming language and was designed for maximum flexibility. As a result of the modular design, the structure can be broken into several independent components. An overview of each of these components appears in the following Section 3.1 with a more detailed discussion of each in sections 3.2 through 3.4.

The Matlab Graphical User Interface Development Environment (GUIDE) toolbox was used to generate the base code for the three main GUI components. Code generated using GUIDE contains a main function that executes the create function and processes the user inputs. A callback function is provided for each object (button, drop down menu, list box, edit box, slider bar, etc).

The layout function creates the GUI from standard Matlab components. A create function is also provided for each object. The callback functions allow the programmer to create code that will execute each time that a button is pressed, text is selected, or information entered. The callback functions allow for the programmer to create code that will execute when an object is created. The create functions were not used for this project.

3.1.1 Program Components

The program can be broken down into three basic components, the graphical user interface, the Simulink models, and the simulation scripts that connect them. The Matlab workspace serves as a common location for stored data. In virtually every location where one subprocess interfaces with another, variables are stored in the Matlab workspace by one process and called by another. This greatly simplifies the calling of functions as variables do not need to be explicitly passed. The simplified flow of information between the basic program components is shown in Figure 3.1.

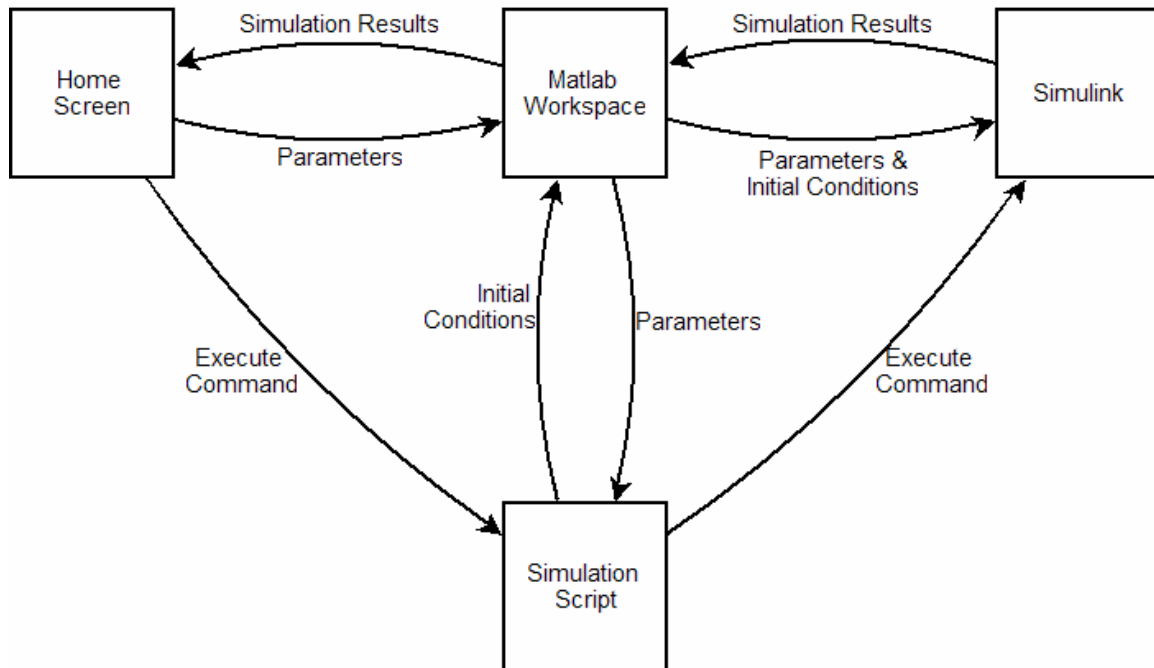


Figure 3.1 Simplified Flow of Information

3.1.1.1 Basic Algorithm

The basic algorithm is shown in Figure 3.2. After the initial program execution, the opening window appears. The user selects a model and presses OK. The home screen appears and the user imports test data and chooses to either batch simulate (run several simulations at once with slight adjustments to one parameter) or to edit parameters himself and simulate. The results are plotted with the test data and the user compares the results. If they are satisfactory, he exits the program, if they are not, he can choose to edit parameters or run more batch simulations. A more detailed version of this flowchart appears in Appendix D.

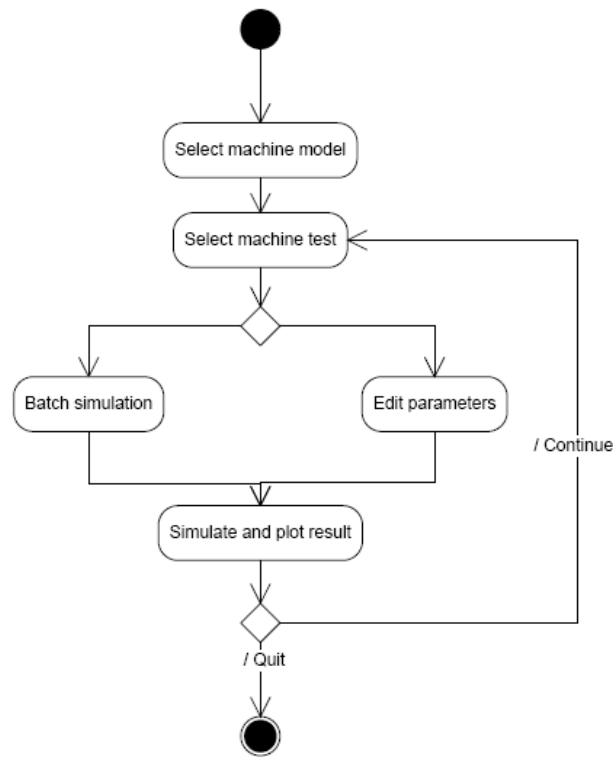


Figure 3.2 Simplified Flow Chart

3.1.1.2 Graphical User Interface

The graphical user interface is intended to provide the user with an easy means of entering parameters and viewing selected simulation results. It also provides several features such as the ability to undo or redo certain actions and import or export data that make it much more user-friendly. It can be broken into three basic components; the opening window (main_window.m), the home window (GUI.m), and the batch simulation window (batch_simulation.m).

The basic purpose of the graphical user interface is to collect parameters from the user, store them in the Matlab workspace, execute the simulation script, collect the arrays of simulation data, and plot them.

3.1.1.3 Simulink Models

The Simulink models are block diagram representations of the state space equations necessary for time domain simulations. It is very important that all constants be entered in as variables to ensure that they can be easily changed and that they will be changeable from the graphical user interface. Selected parameters are stored as arrays in the Matlab workspace.

The standard practice has been to use embedded Matlab functions for the algebraic equations corresponding to the electrical network and loads. It is very important that

every integral have an initial condition defined. It is not necessary to use Simulink models. Instead, a state space model could be formulated and one of the time domain differential equation solvers could be used to solve the system.

3.1.1.4 Simulation Scripts

The simulation scripts are Matlab m-files that contain the equations necessary to calculate the initial conditions. After all of the initial conditions have been calculated, the Simulink model is executed.

3.1.2 The Matlab Workspace

The Matlab Workspace serves as a swap memory or information repository. Any information that needs to be shared between subprocesses is sent to the base Matlab workspace and then read in by the next process. This is intended to simplify function calling.

3.1.3 Setting Files

Setting files are Matlab scripts that create data structures in the Matlab workspace. They are intended to store default data needed to properly initialize the window without requiring the user to enter it manually each time the program executes. The main window setting file (main_window_setting_file.m) stores a list of the models in the library and the names of their respective setting files.

The several GUI setting files are specific to each model. They contain the information necessary to initialize the home screen including the name of each parameter in the Matlab workspace, the name that is to be displayed on the window, the minimum value of each slider bar, maximum and default values. The GUI setting files also contain the workspace and display names for each of the values that are plotted and the names of each test, or simulation case, that the model can execute. Each model must have its own setting file. While creating setting files can be tedious, this method does allow the user to add models to the library without modifying the base graphical user interface code or creating a newly coded graphical user interface.

3.1.4 Test Data

The program allows data to be plotted from Microsoft Excel spreadsheets. This data must be saved in the .xls format and shall be arranged according to Table 3.1. It is important that the first column correspond to time (in seconds) and the remaining columns correspond to the outputs. There must be one and only one column for each output. If no data is available, then that column should be left as zeros, but never blank. The same format is used for exporting simulation data. A sample of test data appears in Appendix A.

Table 3.1 Sample Test Data Format

Time	Output #1	Output #2	Output #3	Output #4
0	0.1	0.2	0.4	0.8
1	0.2	0.4	0.6	1.6
2	0.3	0.6	0.8	2.4
3	0.4	0.8	1.2	3.2

3.1.5 Interaction Between Components and Flow of Information

The flow of information between sub processes is shown in detail in Figure 3.3.

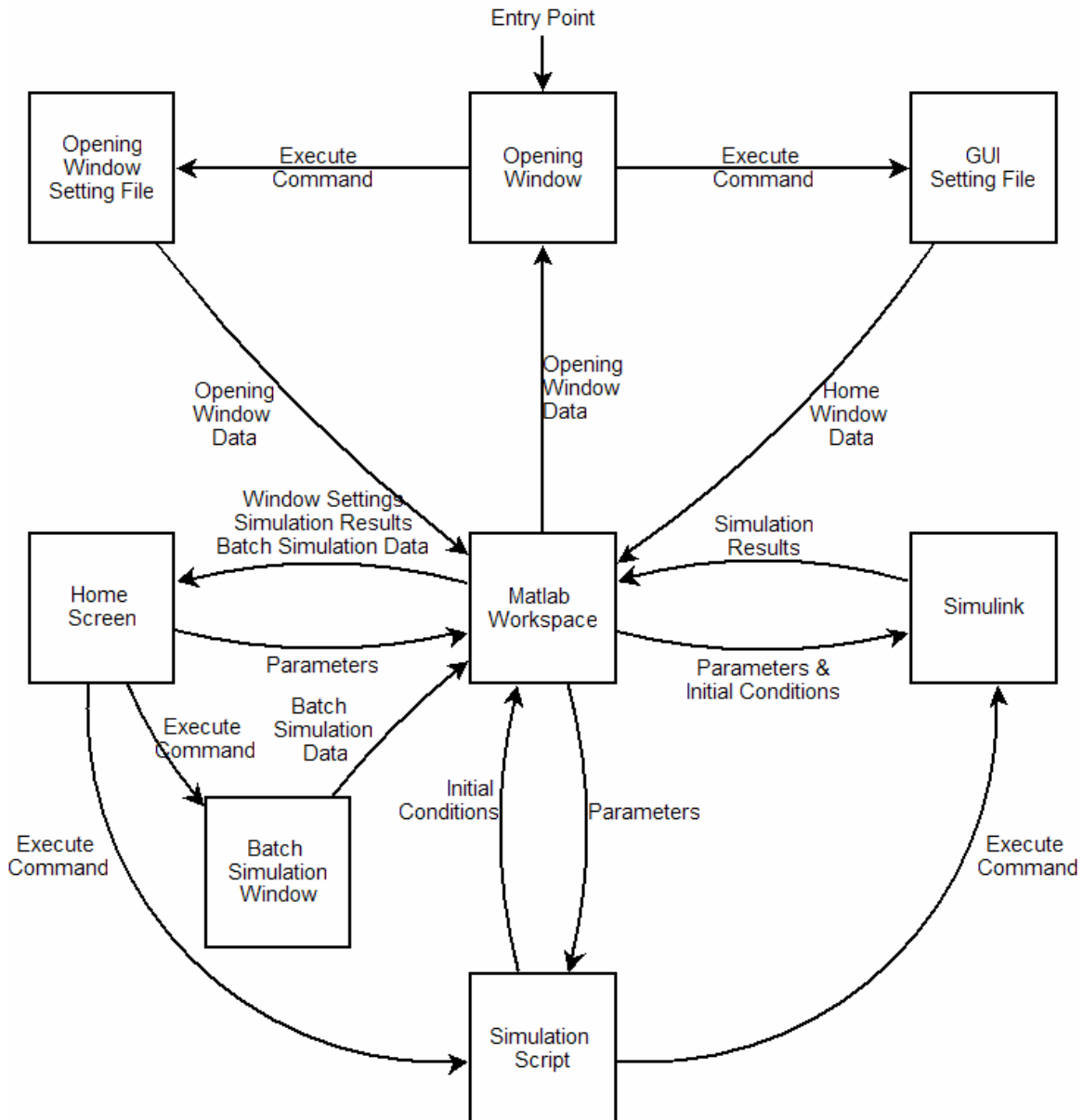


Figure 3.3 Detailed Flow of Information

As one can see, the Matlab workspace serves as the central information storage point. All data that is shared among processes is stored in the Matlab workspace.

3.2 Graphical User Interface

The graphical user interface can be broken into three components: the opening window (main_window.m), home window (GUI.m), and the batch simulation window (batch_simulation.m). Each of these components is described in detail in sections

3.2.2-3.2.4. A discussion on the data structures that store all of the information necessary appears in Section 3.2.1.

3.2.1 Data Structures

Several data structures are used throughout the program, a summary of these appears in Table 3.2. The `model_setting` data structure is used by the opening window. There is one data structure for the entire library. The `parameter_setting` data structure contains the information needed for the parameters panel of the home screen. There is one `parameter_setting` structure for each model. The `output_setting` structure contains the information necessary for the output panel of the home screen. There is one `output_setting` for each model. The `tests_setting` structure is again model specific and contains the information necessary for the test selection menu on the home screen.

Table 3.2 Data Structures and their Fields

Structure	Purpose	Field	Purpose
model_setting	Stores the data needed by the opening window; the name of each model in the library and the setting file associated with that model. There is one structure for the entire library and one entry for each model. Found in main window setting file.	model_name	String value of the mode's name as it should be displayed on the home window.
		setting_file_name	String value of the setting file associated with that particular model.
parameter_setting	Stores the data that is needed for the parameters panel on the home window. This information is also used by the batch simulation window. There is one structure for each model and one entry for each parameter. Found in GUI setting file.	workspace_name	String containing the name of the parameter as it will appear in the workspace and will be used in the simulation script and Simulink model.
		min_value	Double containing the minimum parameter value.
		max_value	Double containing the maximum parameter value.
		default_value	Double containing the default parameter value.
		previous_value	Double containing the previous parameter value. Used for undo/redo functionality.
		current_value	Double containing the parameters current value.
		display_name	String containing the name of the parameter as it will appear on the parameters panel.
		description	Not used on current versions.
		color	A three double array that determines the color of the edit box.
		slider_panel	Not used on current versions.
		editability	An integer that determines for which tests the parameter is active. If this is set to 0 the parameter is always editable, if it is set to a number greater than 0, it will only be editable if the test pulldown menu is set to that position.

Table 3.2 Continued

Structure	Purpose	Field	Purpose
output_setting	Stores the data that is needed for the output panel on the home window. There is one structure for each model and one entry for each output. Found in the GUI setting file.	workspace_name	String containing the name of the variable as it appears in the Simulink block diagram and workspace.
		start_time	Not used on current versions.
		end_time	Not used on current versions.
		display_name	String containing the name that is displayed on the graph's Y-
		test_data	Double array containing the data imported from the Excel spreadsheets.
		simulation_data	Double array containing the simulation data.
		old_simulation_data	Double array containing the data from the previous simulation.
tests_setting	Stores the strings that are needed for the test selection menu on the parameter panel of the home window. Found in the GUI setting file.	test_time	Double array containing the time from the imported Excel data.
		test_name	String containing the name of the test as it is displayed in the test selection pulldown menu.

3.2.2 Opening Window

The opening window is shown in Figure 3.4 and serves as a user friendly interface for model selection. Upon execution, the main window setting file (main_window_setting_file.m) executes, placing the model_setting data structure into the Matlab workspace. The model_setting data structure contains the names of each model and the name of the corresponding GUI setting file. After this executes, a screen similar to Figure 3.4 appears.

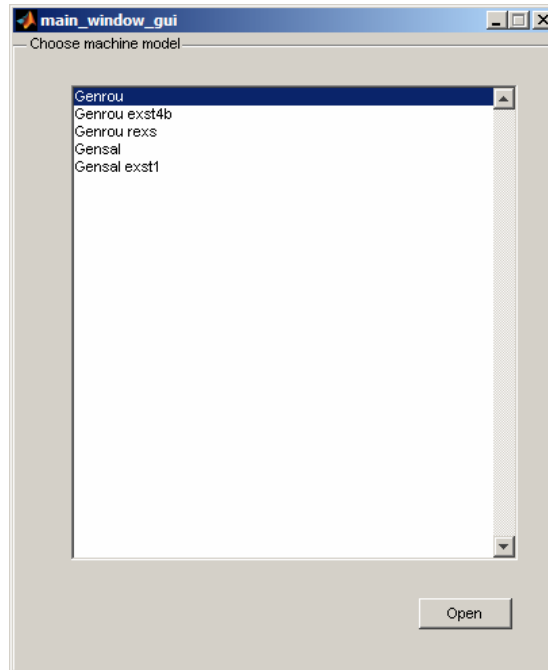


Figure 3.4 Main Window

The program waits for the user to select a model and press the “Open” button. After the “Open” button is pressed, the program executes the GUI setting file that corresponds to the selected model executes, placing the `parameter_setting`, `output_setting`, and `tests_setting` data structures into the workspace. The GUI setting file is described in more detail in Section 3.2.3.2.

3.2.2.1 Structure

The base code for the main window was generated by the Matlab GUIDE toolbox. The create function contains code that executes the main window setting file, placing the `model_setting` structure in the Matlab workspace. This data is then read into a global variable that is used in the layout and “Open” button callback functions. The callback function associated with the “Open” button was modified to execute the simulation script corresponding to the selected model. The main window can support any number of models.

3.2.2.2 Main Window Setting File

The main window setting file is a script file that creates a data structure named `model_setting` and places it in the Matlab workspace. This data structure contains two fields for each entry, one for the model’s name (`model_name`) and the other for the name of the GUI setting file that corresponds to the model.

3.2.3 Home Window

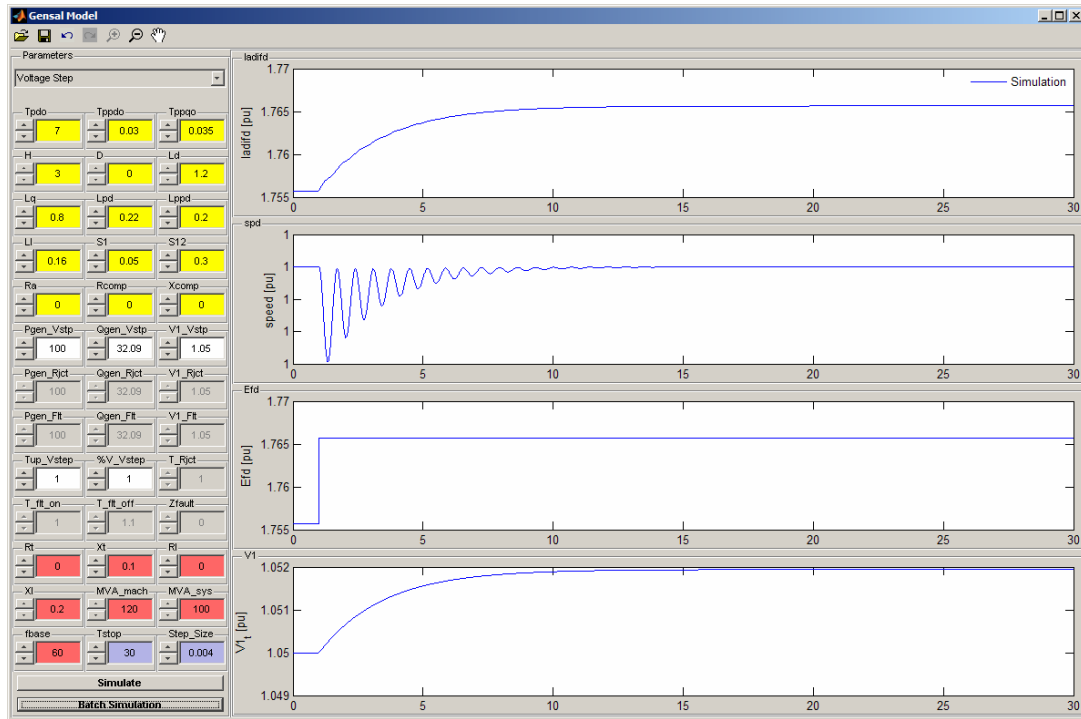


Figure 3.5 Home Window

The home window (GUI.m) is shown in Figure 3.5. This is the main interface that collects parameters from the user and displays the graphs of various parameters versus time. This window can, in general, be divided into three separate panels. The first is the toolbar that appears across the top portion of the display. This panel provides many features that are intended to make the software more user friendly. These are discussed in Table 3.3 in Section 3.2.3.1.1. The code necessary to implement these features is discussed in Section 3.2.3.1.2.

The panel on the left provides space for the simulation parameters while the panel on the right provides space for a graph of each output variable versus time. The amount of space allotted to each sub panel varies depending on the number of parameters displayed. For every 16 parameters, an additional column of parameters is added and an additional 6.8 percent of the display is allocated to parameters.

For each parameter, a sub panel consisting of an edit box and slider bar is provided. The edit box consumes approximately 70 percent of the sub panel while the slider bar consumes the remaining 30 percent. Under normal display resolutions, the slider bar is small enough that only the up and down (increase/decrease) arrows are visible. For each slider bar, there are default, minimum, maximum, and current values, which are found in the parameter_setting data structure. The initial values of these properties are contained in the GUI setting file and, with the exception of the default value, may change through out the session. If a value is entered into the edit box that is not between the minimum and maximum, the range resets to include the new value.

Other important features in the parameters panel are the simulate and batch simulate buttons and the test selection drop down menu. The simulate button places the

parameters in the workspace, executes the simulation script, collects, and plots the simulation results. The batch simulate button allows the user to run several simulations at once, with one variable being changed each time. This allows the user to visualize the effects of one parameter on the simulation results.

The test selection menu allows for several different events to be incorporated into one model and display. (The models built thus far have three different tests; a step change in reference voltage, a “load rejection” test in which the main breakers are opened, and a three phase fault.) Each time that the simulate button is pressed, the entry selected in the test menu is placed into the workspace as an integer, this is then collected by the simulation script and a switch statement determines which initial conditions to use and test to run. Because in real world testing situations, different tests may require different initial conditions, the graphical user interface can make the edit box and slider bar associated with certain parameters inactive based on the test selected in the drop down menu. This is intended to reduce the possibility of entering a parameter in the wrong edit box.

A sub panel is also provided for each output variable. Each sub panel has one plot box that consumes approximately 85 percent of the horizontal space and 92 percent of the vertical space. The sub panels are arranged in a stacked manner, with the horizontal (x) axis being much longer than the vertical (y) axis in any case where more than one graph is displayed.

3.2.3.1 Structure

A great deal of effort has been placed in to creating the entire graphical user interface as flexible and easily expandable as possible. The home screen has been created in such a way so as to in theory allow an infinite number parameters and plots, however, due to space restrictions, the user should attempt to limit the number of parameters to 96. The base code for the graphical user interface was created using GUIDE by building a basic window with the toolbar, parameters panel, and graph panel and then modifying the code to allow an infinite number of number of parameters and graphs.

Upon execution, the parameter_setting, model_setting, and tests_setting data structures are collected from the Matlab workspace and are saved as global variables. These are used throughout the program. After the main window appears the interface waits for user commands. After the simulate button is pressed, each parameter is placed in the Matlab workspace as a double variable. An integer corresponding to the selected test is placed in the workspace. The simulation script is executed, after this, the simulation data is collected from the Matlab workspace and plotted. Because the exact number of parameters or outputs is not fixed, loops were used. A wait bar (see Figure 3.6) is provided to remind the user that the simulation is in process. To ensure that the wait bar remains on top of all other Matlab windows, the window should be modal.

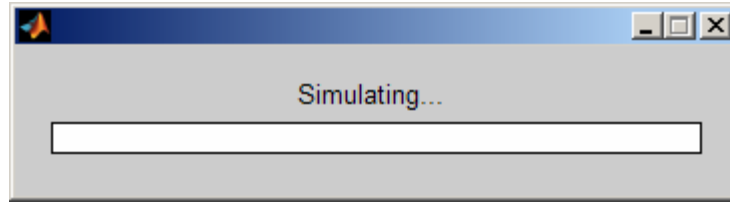


Figure 3.6 Wait bar

3.2.3.1.1 Features

Several user friendly features have been added. A summary of these features and their functionality appears in Table 3.3.

Table 3.3 Product Requirements and Features

Feature	Required or Supplemental	Intended Functionality
Edit Box	Required	Provide the user with a means to quickly enter a specific value for a parameter.
Slider Bar	Required	Provide the user with an easy method to vary parameters. Each slider bar shall have a default maximum and minimum value. When the slider bar position is changed, the value in the edit box shall update. If a value is entered into the edit box that is outside of this range, the corresponding maximum or minimum should reset to the value in the edit box.
Interface with Simulink	Required	The user shall be able to execute Simulink models from the graphical user interface.
Graph Simulation Data	Required	Simulation data shall be plotted on figures in the graphical user interface.
Import Data from Excel and Display on Graphs	Required	The user shall be able to import time domain data stored in Microsoft Excel spreadsheets in a format specified by the project team and plot them on the same graphs as the simulation data.
Undo/Redo Actions	Required	The user shall be able to undo actions upto to previous simulation.
Save Current Session	Required	The user shall be able to save the position of each slider bar.
Load Previous Session	Required	The user shall be able to load the position of each slider bar from a previous session.
Zoom In and Out on Graphs	Required	The user shall be able to zoom in and out on all graphs.

Table 3.3 Continued.

Feature	Required or Supplemental	Intended Functionality
Handle Simulation of Different Tests	Required	The graphical user interface shall be able to handle models capable of simulating different events
Handle Multiple Models	Required	The graphical user interface shall be capable of handling a library of any number of models. The project team shall enter at least 5 models into this library.
Pan Graphs	Supplemental	The user should be able to pan graphs.
Save Graphs as Matlab Figures	Supplemental	The user should be able to save graphs as Matlab figures.
Export Simulation Data to Excel Spreadsheet	Supplemental	The user should be able to save simulation data as an Excel spreadsheet in a format that is the same as the imported data.
Batch Simulation	Supplemental	The user should be able to choose a single parameter, specify a minimum and maximum value, and any number of simulations and have the program iterate that many times between those two points and graph the simulation data.
Hold Previous Zoom Level	Supplemental	When test data has been imported, the Graph's zoom level should not change after the simulate button is pressed.
Graph Previous Simulation Data	Supplemental	The previous simulation data should appear as a light gray line to allow the user to judge progress in matching the test data.
Color Code for Edit Box	Supplemental	Color edit boxes to help user to visually group different types of parameters belonging to different subsystems.

3.2.3.1.2 Callback Functions

Callback functions are used to specify what actions should occur after a certain action is performed. The callback functions necessary for the completion of the main window appear in Sections 3.2.3.1.2.1 through 3.2.3.1.2.8.

3.2.3.1.2.1 Slider Bar and Edit Box Callback Functions

The slider bars share a common callback function. (The edit boxes do as well.) This is possible because each slider bar and edit box has its own handle, a double identification number that is unique to that object. The remaining buttons have their own callback functions. The edit box callback function collects the new value from the

edit box and changes the position of the slider bar. The slider bar's range is adjusted if necessary. The slider bar call back function contains code that updates the value in the edit box and adjusts the position of the slider bar.

3.2.3.1.2.2 Test Selection Menu Callback Function

The test selection drop down menu call back function first deletes both the current and old simulation data so that the gray trace associated with the previous simulation will not appear when the simulation button is next pressed. Each parameter is then checked to see if its status should be changed from active to inactive or vice versa.

3.2.3.1.2.3 Simulate Button Callback Function

The simulate button callback function first makes the simulate button inactive and changes the text from "Simulate" to "Simulating..." Next a wait bar is created and each parameter is placed in the workspace. If this test has been simulated previously, the old simulation data is placed into the old_simulation_data field in the output_setting data structure and the simulation is executed. If there is a problem with the simulation, a catch statement will cause a message box displaying "Simulation error" to appear. This is shown in Figure 3.7. This prevents the program from crashing, but does not provide much insight into the source of the problem. If the simulation executes properly, the results are plotted and a legend appears in the upper right hand corner of the top plot box. A custom color scheme is defined to ensure that all of the plots are legible and that all colors are pleasing to the eye.

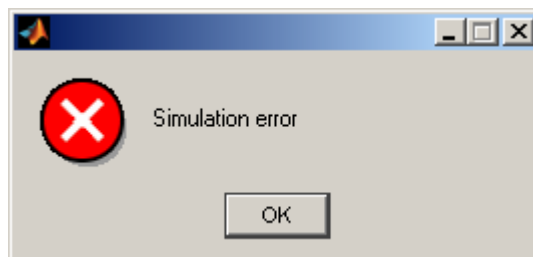


Figure 3.7 Simulation Error Message Box

3.2.3.1.2.4 Batch Simulate Button Callback Function

The batch simulate call back function starts by making the batch simulate button inactive and changing the text from "Batch Simulate" to "Simulating..." The batch simulation window (See Figure 3.8) appears and the program suspends execution until the user presses either the OK or cancel button. After this, the program resumes execution, if the cancel button was pressed, the program returns to the main function and does not simulate. If the OK button was pressed, a wait bar is displayed and the function collects the parameter, minimum value, maximum value, and number of simulations from the workspace. After this an array of parameter values for the parameter that will be varied is created and all of the parameters are placed in the workspace. After the simulation has completed, the data is plotted. The program repeats this for the desired number of simulations.

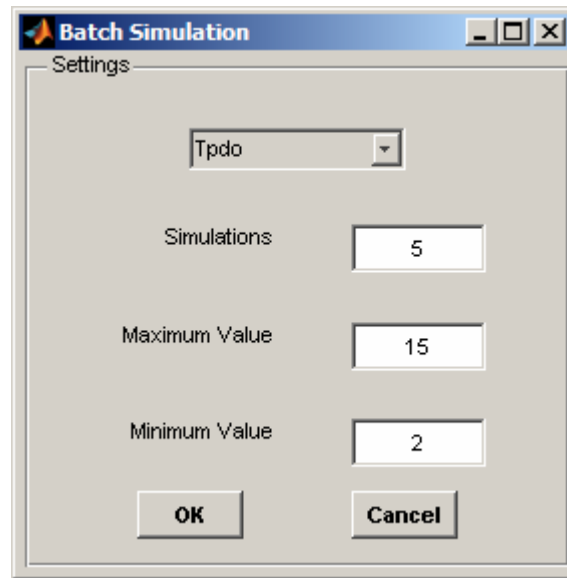


Figure 3.8 Batch Simulate Window

3.2.3.1.2.5 Open Button Callback Function

The open button callback function starts by opening the standard windows “Open File” window. This window is shown in Figure 3.9. The user has the option to either import test data from an Excel spreadsheet or load a saved case in the form of a Matlab .mat file.

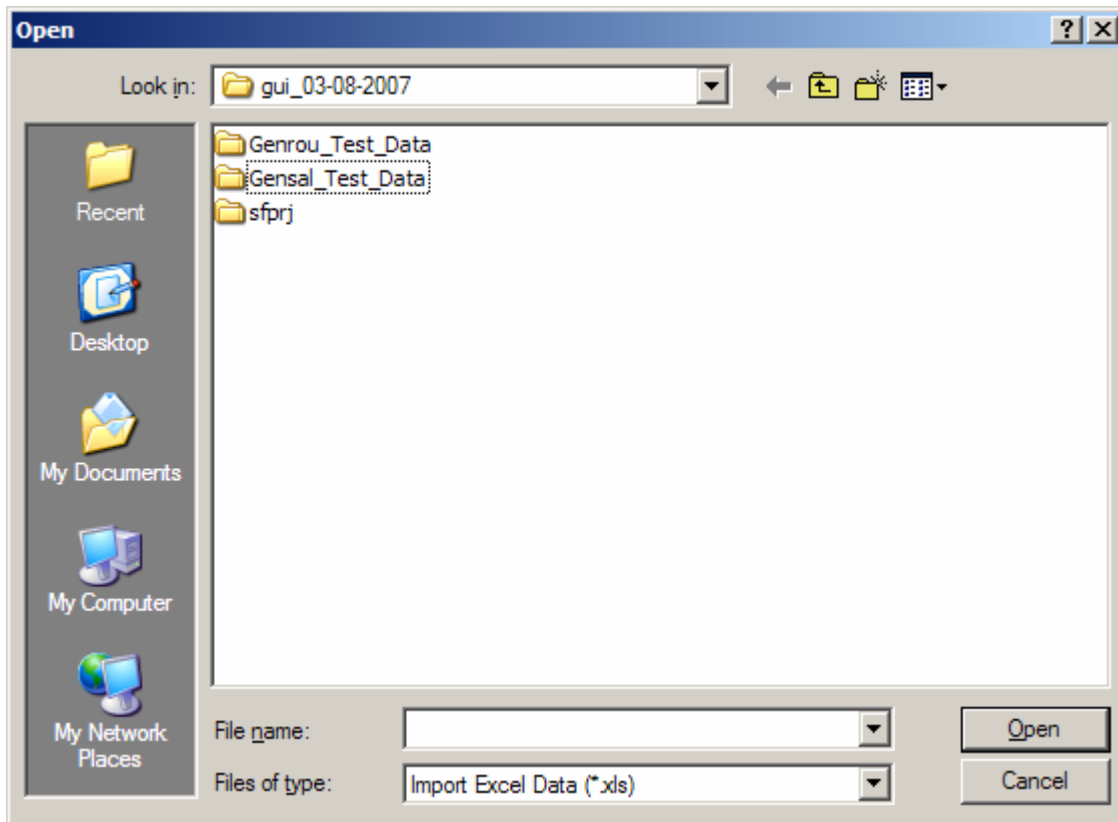


Figure 3.9 Open Window

After the user has selected a file, the program attempts to open it as an Excel file, if it succeeds, the data is plotted, if it fails, the program attempts to open it as a .mat save case. If this succeeds, each parameter is set to the value in the file, if this fails as well, an error message is displayed.

3.2.3.1.2.6 Save Button Callback Function

The save button callback function starts by opening the standard Windows Save As dialog box. This is shown in Figure 3.10. The user has the option of saving three different files. The first is to save the output plots as Matlab figures (.fig). If this option is chosen, the program will save each graph individually with the workspace name of the output variable concatenated to the end of the filename. If the user elects to save the parameter data (save case) as a .mat file, the parameter_setting array is saved. If the user wishes to save the simulation data, the simulation data is saved to an Excel spreadsheet in the format in Section 3.1.4.

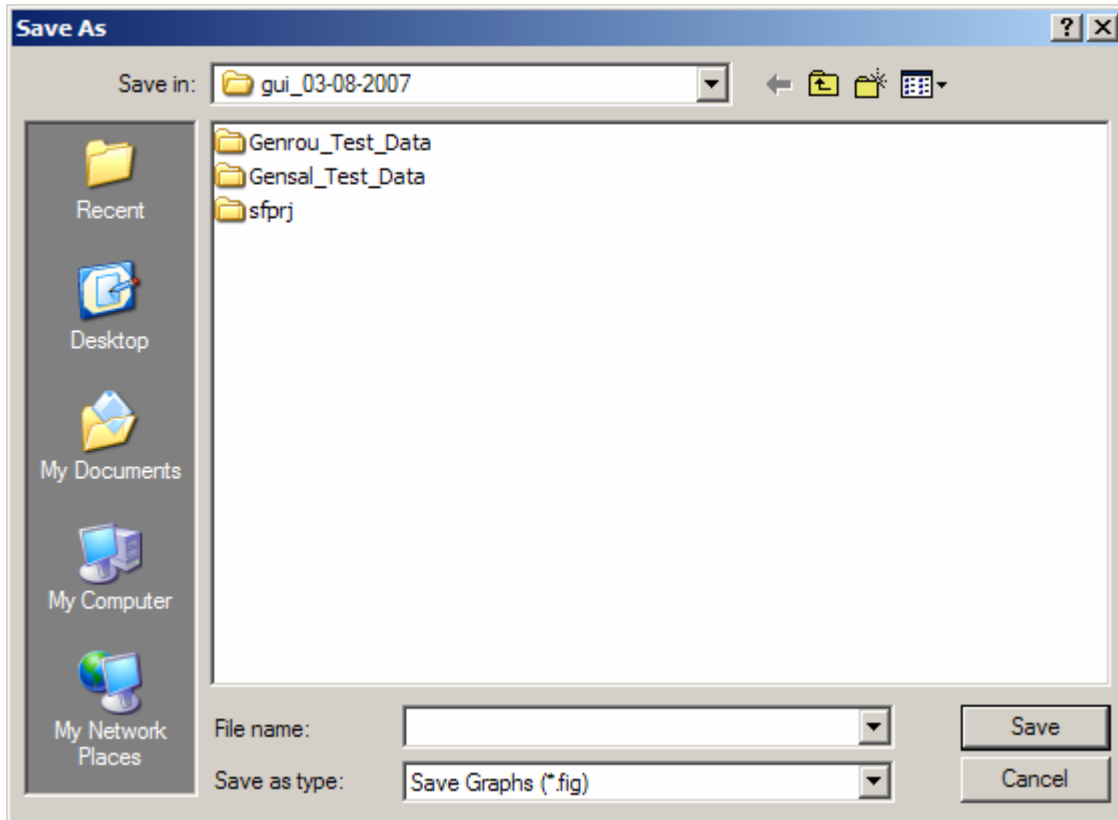


Figure 3.10 Save As Window

3.2.3.1.2.7 Undo/Redo Button Callback Functions

If the Undo toolbar button is pressed, the program changes the last modified parameter to its previous value and enables the redo button. If the redo button is pressed, the last undone parameter is changed to its pre-undo value. It should be noted that the undo functionality is only available after the simulate button has been pressed and can only undo changes up to the previous simulation.

3.2.3.1.2.8 Zoom In, Zoom Out, and Pan Button Callback Functions

The zoom in, zoom out, and pan callback functions are all very similar. Each one changes the zoom property of each graph to either zoom in, zoom out or pan, disables the button that was pressed, and enables the other two.

3.2.3.2 GUI Setting File

The GUI setting file is a Matlab script that creates three data structures. The first, `parameter_setting`, contains the data necessary for the slider bars and edit boxes. There is one entry in this data structure for each slider bar or edit box on the parameters panel. The next, `output_setting`, contains all of the information necessary for the graphs on the output panel. The last, `test_setting`, contains the information needed for the test selection menu. These data structures are described in detail in Table 3.2. Three additional string variables, `simulin_file_name`, `simulation_file`, and `simulation_script`, are also placed in the workspace. `simulin_file_name` contains the

name of the Simulink block diagram. `simulation_file` contains the title given to the home window. `simulation_script` contains the name of the simulation script associated with that particular model. An example setting file is included in Appendix B.

3.2.4 Batch Simulation Window

The batch simulation window, shown in Figure 3.7 allows users to run several simulations at one time, varying one parameter each time. This is intended to allow the user to see the effects of different values of one parameter and to aid in the overall parameter selection process.

When the batch simulate button is pressed, the batch simulate window appears. A pull down menu allows the user to select which parameter will be varied. The maximum and minimum values correspond to the points between which the program will iterate. By default, these are set at the minimum and maximum slider bar values. A callback function ensures that when the position of the pull down menu changes, that the maximum and minimum values will change. The simulations edit box allows the user to set the number of times that the program will run. This number must be an integer greater than 0.

When the OK button is pressed five variables are passed to the Matlab workspace. `parameter`, is an integer corresponding to the parameter selected in the pull down menu. The maximum value, `max`, the minimum value, `min`, the number of simulations, `simulations`, and an exit flag `cancel`. If `cancel` is set to 1 because the cancel button was pressed the home screen will not run the simulation, if the OK button is pressed, `cancel` will be set to 0 and the home screen will run the simulations.

3.3 Simulation Scripts

The simulation scripts are Matlab scripts that collect parameters from the workspace, determine which test to run, and initial conditions to use, calculates additional initial conditions, and executes the Simulink model.

Though these scripts are very flexible, the standard practice has been to use a switch statement that switches based on the test selected. The initial conditions corresponding to this test are chosen in this statement. In the lines that follow, more initial conditions are calculated using algebraic expressions. The last line executes the Simulink model. It is very important that the variables used in the simulation script exactly match the variables used in the Simulink models and GUI setting files.

It should be noted that it is not necessary to use Simulink block diagrams. The simulation script can be used to build a state space model of the system being studied. Any one of the Matlab differential equation solvers can be used to perform the simulation.

3.4 Simulink Models

The Simulink models used for simulations are also very flexible. They can be models of any dynamic system and do not need to be from the power industry. The standard practice has been to use block diagrams for the generator windings, exciters, voltage

regulators, turbines, and governors and imbedded Matlab code for the algebraic equations corresponding to the electrical network.

The simulation time and step size need to be adjustable via the home screen, thus the appropriate settings in the configuration parameters dialog box (Simulation -> Configuration Parameters) need to be changed to variables. Variables will be used in place of constants on all blocks in the algebraic equations in order to maximize flexibility. A sample Simulink block diagram appears in Appendix C.

4 Resources and Schedules

This section gives a detailed schedule that was followed by the team throughout the project. It also contains the resources that were used during the course of the project. It includes the original schedule and resource estimates predicted for the project as well as the actual usage for the course of the project.

4.1 Resource Requirements

The resources that were required for the project completion were mainly time and expertise provided by the team members, advisor and the client. Other resources such as computer hardware and software were provided by the University with the exception of poster board, lamination, document binding, and shipping.

4.1.1 Personnel Effort Requirements

This section shows the chart representing man hours based on our Gantt chart schedule. Table 4.1 shows our original estimates and Table 4.2 shows the actual usage for the entire project. The only major differences between the original estimates and the actual are the allocations for design and implementation. We have adjusted the numbers showing that implementation took more time than originally estimated.

Table 4.1 Original Labor Estimates

Task	Mark	Adam	Jared	Yu	Task Total
Task 1 - Problem Definition	4	4	2	4	14
<i>Task 1a - Problem Definition Completion</i>	4	2	0	0	6
<i>Task 1b - End Users and End Uses Identification</i>	0	0	2	2	4
<i>Task 1c - Constraint Identification</i>	0	2	0	2	4
Task 2 - Technology and Implementation Considerations and Selection	3	4	2	3	12
<i>Task 2a - Identification of Possible Technologies</i>	0	2	0	2	4
<i>Task 2b - Identification of Selection Criteria</i>	1	0	1	0	2
<i>Task 2c - Technology Research</i>	2	2	0	0	4
<i>Task 2d - Technology Selection</i>	0	0	1	1	2
Task 3 - Prototype Design	22	28	23	31	104
<i>Task 3a - Identification of Design Requirements</i>	1	3	0	4	8
<i>Task 3b - Design Process</i>	15	15	11	17	58
<i>Task 3c - Documentation of Design</i>	6	10	12	10	38
Task 4 - Prototype Implementation	23	18	16	23	80
<i>Task 4a - Implement Prototype Block Diagram</i>	10	11	9	10	40
<i>Task 4b - Implement Prototype User Interface</i>	13	7	7	13	40
Task 5 - Prototype Testing	11	18	20	11	60
<i>Task 5a - Test Planning</i>	2	4	2	0	8
<i>Task 5b - Test Development</i>	0	2	0	4	6

Task	Mark	Adam	Jared	Yu	Task Total
<i>Task 5c - Test Execution</i>	2	4	2	2	10
<i>Task 5d - Test Evaluation</i>	0	0	4	2	6
<i>Task 5e - Documentation of Testing</i>	7	8	12	3	30
Task 6 – Feedback	0	1	3	8	12
<i>Task 6a - Client Feedback</i>	0	0	0	8	8
<i>Task 6b - Feedback Implementation</i>	0	1	3	0	4
Task 7 - End Product Design	24	29	29	20	102
<i>Task 7a - Identification of Design Requirements</i>	1	0	3	0	4
<i>Task 7b - Design Process</i>	15	20	12	13	60
<i>Task 7c - Documentation of Design</i>	8	9	14	7	38
Task 8 - End Product Implementation	18	19	22	21	80
<i>Task 8a - Implement End Product Block Diagram</i>	13	5	17	5	40
<i>Task 8b - Implement End Product User Interface</i>	5	14	5	16	40
Task 9 - End Product Testing	16	18	14	14	62
<i>Task 9a - Test Planning</i>	2	2	4	0	8
<i>Task 9b - Test Development</i>	0	2	2	4	8
<i>Task 9c - Test Execution</i>	5	2	0	3	10
<i>Task 9d - Test Evaluation</i>	4	2	0	0	6
<i>Task 9e - Documentation of Testing</i>	5	10	8	7	30
Task 10 - End Product Documentation	5	6	9	2	22
<i>Task 10a - Development of End User Documentation</i>	5	6	9	2	22
Task 11 - End Product Demonstration	9	8	5	4	26
<i>Task 11a - Demonstration Planning</i>	7	6	4	3	20
<i>Task 11b - Faculty Advisor(s) Demonstration</i>	0	1	0	1	2
<i>Task 11c - Client Demonstration</i>	1	1	0	0	2
<i>Task 11d - Industrial Review panel Demonstration</i>	1	0	1	0	2
Task 12 - Project Reporting	25	33	19	16	93
<i>Task 12a – Project Plan Development</i>	8	10	6	9	33
<i>Task 12b - Design Report Development</i>	4	6	9	1	20
<i>Task 12c - Project Poster Development</i>	8	8	0	4	20
<i>Task 12d - Project Final Report Development</i>	5	9	4	2	20
Project Total	160	186	164	157	667

Table 4.2 Actual Labor Usage

Task	Mark	Adam	Jared	Yu	Task Total
Task 1 - Problem Definition	1	2	1	1	5
<i>Task 1a - Problem Definition Completion</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>	<i>0</i>
<i>Task 1b - End Users and End Uses Identification</i>	<i>1</i>	<i>0</i>	<i>1</i>	<i>0</i>	<i>2</i>
<i>Task 1c - Constraint Identification</i>	<i>0</i>	<i>2</i>	<i>0</i>	<i>1</i>	<i>3</i>
Task 2 - Technology and Implementation Considerations and Selection	4	2	3	4	13
<i>Task 2a - Identification of Possible Technologies</i>	<i>0</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
<i>Task 2b - Identification of Selection Criteria</i>	<i>1</i>	<i>0</i>	<i>1</i>	<i>0</i>	<i>2</i>
<i>Task 2c - Technology Research</i>	<i>2</i>	<i>2</i>	<i>0</i>	<i>0</i>	<i>4</i>
<i>Task 2d - Technology Selection</i>	<i>1</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>4</i>
Task 3 - Prototype Design	19	21	16	21	77
<i>Task 3a - Identification of Design Requirements</i>	<i>1</i>	<i>2</i>	<i>0</i>	<i>2</i>	<i>5</i>
<i>Task 3b - Design Process</i>	<i>15</i>	<i>19</i>	<i>8</i>	<i>17</i>	<i>59</i>
<i>Task 3c - Documentation of Design</i>	<i>3</i>	<i>0</i>	<i>8</i>	<i>2</i>	<i>13</i>
Task 4 - Prototype Implementation	39	37	46	43	165
<i>Task 4a - Implement Prototype Block Diagram</i>	<i>21</i>	<i>21</i>	<i>30</i>	<i>8</i>	<i>80</i>
<i>Task 4b - Implement Prototype User Interface</i>	<i>18</i>	<i>16</i>	<i>16</i>	<i>35</i>	<i>85</i>
Task 5 - Prototype Testing	5	6	12	10	33
<i>Task 5a - Test Planning</i>	<i>3</i>	<i>1</i>	<i>1</i>	<i>5</i>	<i>10</i>
<i>Task 5b - Test Development</i>	<i>0</i>	<i>4</i>	<i>0</i>	<i>1</i>	<i>5</i>
<i>Task 5c - Test Execution</i>	<i>1</i>	<i>1</i>	<i>2</i>	<i>4</i>	<i>8</i>
<i>Task 5d - Test Evaluation</i>	<i>1</i>	<i>0</i>	<i>4</i>	<i>0</i>	<i>5</i>
<i>Task 5e - Documentation of Testing</i>	<i>0</i>	<i>0</i>	<i>5</i>	<i>0</i>	<i>5</i>
Task 6 – Feedback	1	3	5	7	16
<i>Task 6a - Client Feedback</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>4</i>
<i>Task 6b - Feedback Implementation</i>	<i>0</i>	<i>2</i>	<i>4</i>	<i>6</i>	<i>12</i>
Task 7 - End Product Design	17	16	25	19	77
<i>Task 7a - Identification of Design Requirements</i>	<i>1</i>	<i>3</i>	<i>1</i>	<i>3</i>	<i>8</i>
<i>Task 7b - Design Process</i>	<i>15</i>	<i>10</i>	<i>10</i>	<i>15</i>	<i>50</i>
<i>Task 7c - Documentation of Design</i>	<i>1</i>	<i>3</i>	<i>14</i>	<i>1</i>	<i>19</i>
Task 8 - End Product Implementation	45	40	43	41	169
<i>Task 8a - Implement End Product Block Diagram</i>	<i>5</i>	<i>7</i>	<i>11</i>	<i>6</i>	<i>29</i>
<i>Task 8b - Implement End Product User Interface</i>	<i>40</i>	<i>33</i>	<i>32</i>	<i>35</i>	<i>140</i>
Task 9 - End Product Testing	22	6	16	18	62
<i>Task 9a - Test Planning</i>	<i>5</i>	<i>0</i>	<i>6</i>	<i>5</i>	<i>16</i>
<i>Task 9b - Test Development</i>	<i>8</i>	<i>2</i>	<i>2</i>	<i>10</i>	<i>22</i>
<i>Task 9c - Test Execution</i>	<i>5</i>	<i>2</i>	<i>0</i>	<i>3</i>	<i>10</i>
<i>Task 9d - Test Evaluation</i>	<i>0</i>	<i>2</i>	<i>2</i>	<i>0</i>	<i>4</i>
<i>Task 9e - Documentation of Testing</i>	<i>4</i>	<i>0</i>	<i>6</i>	<i>0</i>	<i>10</i>
Task 10 - End Product Documentation Development	9	8	32	4	53

Task	Mark	Adam	Jared	Yu	Task Total
Task 11 - End Product Demonstration	14	25	16	6	61
<i>Task 11a - Demonstration Planning</i>	<i>11</i>	<i>22</i>	<i>12</i>	<i>3</i>	<i>48</i>
<i>Task 11b - Faculty Advisor(s) Demonstration</i>	<i>1</i>	<i>1</i>	<i>2</i>	<i>1</i>	<i>5</i>
<i>Task 11c - Client Demonstration</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>4</i>
<i>Task 11d - Industrial Review panel Demonstration</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>1</i>	<i>4</i>
Task 12 - Project Reporting	46	44	46	19	155
<i>Task 12a – Project Plan Development</i>	<i>4</i>	<i>10</i>	<i>5</i>	<i>4</i>	<i>23</i>
<i>Task 12b - Design Report Development</i>	<i>4</i>	<i>6</i>	<i>9</i>	<i>6</i>	<i>25</i>
<i>Task 12c - Project Poster Development</i>	<i>5</i>	<i>4</i>	<i>11</i>	<i>0</i>	<i>20</i>
<i>Task 12d - Project Final Report Development</i>	<i>33</i>	<i>24</i>	<i>21</i>	<i>9</i>	<i>87</i>
Project Total	222	210	261	193	886
Meetings and Correspondence	75	90	90	60	315
Grand Total	297	300	351	253	1201

4.1.2 Other Resource Requirements

Computers running Microsoft Office, Matlab 2006 and Simulink were required for both programming and documentation through out the project. Server space was also required for sharing files between team members and hosting the project website. Our original estimate for binding was \$25, this was changed to \$72 because there were 4 documents being bound at \$13 each and \$5 shipping to the client per document. Poster printing was done at no charge by the department so our original estimate of \$50 was reduced to actually \$12 for lamination and \$20 for poster board and glue.

Computers required were available in the computer labs provided by the University. The \$104 binding and poster fee was split between team members.

4.1.3 Financial Requirements

This section details the cost associated with the resource requirements. The original estimate of project costs appears in Table 4.3, the revised estimate appears in Table 4.4, the reason for the increased cost is the increase in estimated labor.

Table 4.3 Financial Costs Associated with Resources (Original Estimate)

Resource	Cost
Labor	\$6670
Materials	\$75
Total	\$6750

Table 4.4 Financial Costs Associated with Resources (Actual Costs)

Resource	Cost
Labor	\$12,010
Materials	\$104
Total	\$12,114

4.2 Schedules

This section contains the spring and fall schedules showing original estimates, revised estimates, and actual times. Figure 4.1 shows a legend for the spring and fall charts, Figure 4.2 shows the fall 2006 semester schedule, Figure 4.3 shows the spring 2007 schedule and Figure 4.4 shows the deliverable dates. Table 4.5 shows task dates for the original schedule and Table 4.6 shows the final task dates. The only significant differences between the original and final schedules are the lengths associated with design and implementation of the prototype and end product. The deliverables dates have not been changed so there is only one schedule associated with it.

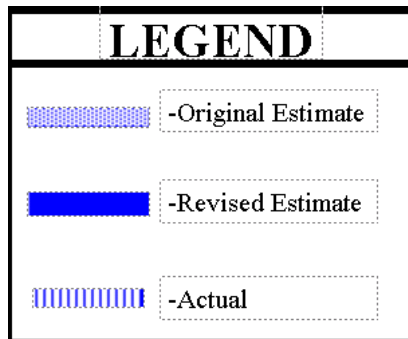


Figure 4.1 GANTT Chart Legend

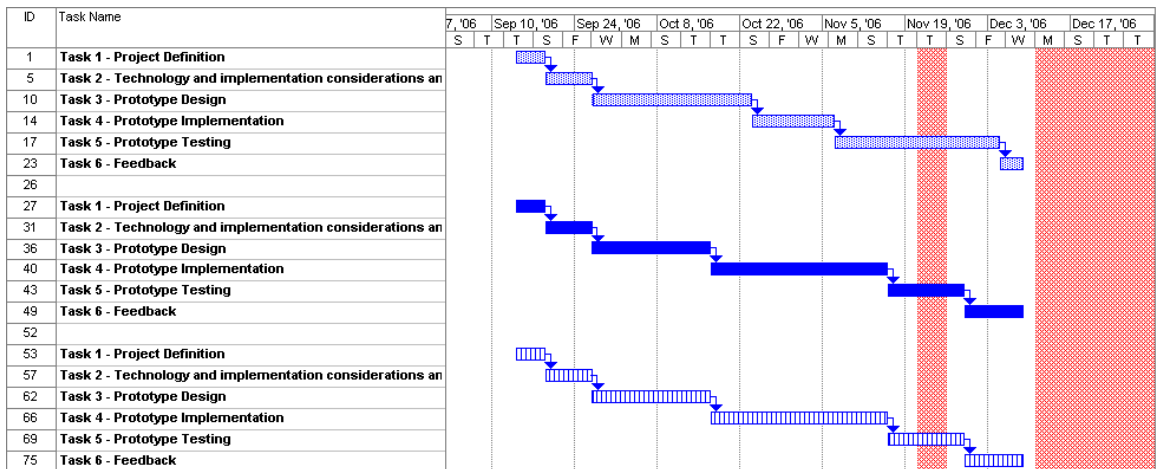


Figure 4.2 Fall 2006 Schedule

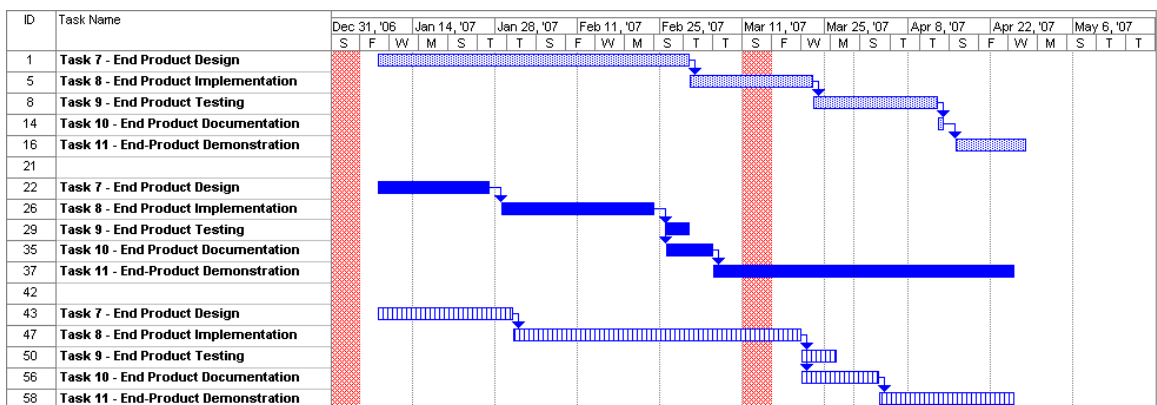


Figure 4.3 Spring 2007 Schedule

5 Closure Materials

This section contains an evaluation of the project in terms of success, commercialization possibility, recommendations for additional work, what lessons were learned and risk management. It also contains a closing summary and the project participants' contact information.

5.1 Project Evaluation

This section explains the degree of success of the project. It shows several main functional requirements and their degree of success. The success of each requirement is evaluated as one of the following: greatly exceeded, exceeded, fully met, partially met, not met, or not attempted. Any feature not evaluated as fully met in Table 5.1 is discussed below. The testing supporting that these features have been properly implemented appears in Section 2.1.5.

Table 5.1 Project Task Evaluation

<i>Feature</i>	<i>Required or Supplemental</i>	<i>Evaluation</i>
Edit Box	Required	Fully met
Slider Bar	Required	Fully met
Interface with Simulink	Required	Fully met
Graph Simulation Data	Required	Fully met
Import Data from Excel and Display on Graphs	Required	Fully met
Undo/Redo Actions	Required	Fully met
Save Current Session	Required	Fully met
Load Previous Session	Required	Fully met
Zoom in and Out on Graphs	Required	Fully met
Pan Graphs	Supplemental	Fully met
Handle Multiple Models	Required	Exceeded
Handle Simulation of Different Tests	Required	Exceeded
Save Graphs as Matlab Figures	Supplemental	Fully met
Export Simulation Data to Excel Spreadsheet	Supplemental	Fully met
Batch Simulation	Supplemental	Exceeded

<i>Feature</i>	<i>Required or Supplemental</i>	<i>Evaluation</i>
Hold Previous Zoom Level	Supplemental	Fully met
Graph Previous Simulation Data	Supplemental	Fully met
Color Code for Edit Box	Supplemental	Fully met
Changed Status for Certain Parameters to Active or Inactive Based on Currently Selected Test	Supplemental	Fully met

The tasks handle multiple models and handle simulation of different tests were both fully completed to the requirements of the project plan. However the team has spent time above and beyond adding additional models and tests. The formulation of setting files for the models and tests allows engineers to include additional models in the future with ease. The batch simulation feature was not only a supplemental component included near the project completion, but it also allows for a small amount of automation for any parameter for any number of simulations. This will allow the engineer to use their time more efficiently. The project as a whole is evaluated as very successful because each requirement was at least fully met, and every supplemental requirement will save the engineer additional time and therefore money for the company.

Additionally the team has provided an assessment of the success of each milestone. This appears in Table 5.2. Overall, the team feels that they were very successful and that all of the milestones were very successful. The team feels that had the client's needs been more clearly determined at an earlier date, the second semester implementation might have been smoother and the research, though well executed, was not fruitful. A great deal of time was spent on implementation and the quality of the documentation suffered as a result. Testing took much longer than anticipated and the end product reviews, though positive, did reveal some improvements that could have been made.

Table 5.2 Milestone Evaluation

Milestone	Relative Importance	Evaluation Score	Resultant Score
Problem Definition	10%	90%	9%
Research	8%	85%	6.8%
Technology Selection	7%	100%	7%
Prototype Design	15%	100%	15%
End-Product Implementation	25%	100%	25%
End-Product Testing	15%	90%	13.5%
End-Product Documentation	5%	90%	4.5%
Project Reviews	7%	85%	5.95%
End-Product Demo	8%	100%	8%
Overall	100%		94.75%

5.2 Commercialization Possibility

This program is very flexible and has uses in virtually every case where the parameters needed for accurate computer modeling are difficult to determine. The project team purposely did not make any decisions that would limit the program to any one field. However, because of licensing difficulties associated with Matlab and Simulink, the client has opted not to commercialize this product at any level.

5.3 Recommendations for Additional Work

There are several features that were outside the scope of this project. A wizard for adding new generator models is one such feature. Another feature that could be created that would take significant work is an algorithmic parameter analyzer that would somewhat automate the process of machine parameter determination.

5.4 Lessons Learned

This section details a few of the things the team has learned during the course of the last two semesters.

5.4.1 What Went Well

The team learned a great deal working on this project. Things that went very well were the streamlining of the process of machine model entering once the prototype file structure was created. During the second semester the team received a great deal of information from the client that allowed us to quickly and consistently add features and root out bugs in the system.

5.4.2 What Did Not Go Well

The things that did not go well were our inability to create machine models on our own in the beginning of the project. Developing the models from scratch is normally considered advanced graduate material and was simply outside the realm of the project team's understanding.

5.4.3 What Technical Knowledge Was Gained

The team gained a great deal of knowledge about the functions of Matlab and how to create programs in the environment. The team also learned how to use different parts of Matlab to pass information back and forth for data analysis. The project team also gained a great deal of understanding of the considerations of machine operation, testing, and modeling. Power systems engineers and professors were consistently very impressed with both the difficulty of the project and the depth of understanding that team members had for complex and difficult material.

5.4.4 What Non-Technical Knowledge Was Gained

Team relationships proved to be a major struggle for our team but offered a powerful learning environment for interpersonal relationships in a team-oriented technical project.

5.4.5 What The Team Would Do Differently If The Team Had To Do It Over Again

If given the opportunity to start again with the knowledge the team have now, the team would have spent more time in the beginning of the project gathering requirements, interfacing with the client, and determining what they really needed. This may have alleviated some confusion about program functionality and aesthetics and would have eliminated some of the last minute feature additions.

5.5 Risk and Risk Management

This section lists anticipated and unanticipated risks and some of the mitigation techniques used to minimize the risk.

5.5.1 Anticipated potential risks and planned management

The fact that this is a software project alleviated many of the common concerns regarding ordering of parts and what to do if expensive parts were damaged or important parts were not available.

The biggest risk management issue that the team faced was eliminating the possibility losing important data. Because our entire project is a few files, it was very important that the team back them up in multiple places. The group set up an FTP server that was used not only to share data amongst group members, but also to ensure that multiple copies of all files were maintained. This server was backed up several times a week.

5.5.2 Anticipated risks encountered and success in management

Fortunately, data loss did not pose a problem at any point during the project and no files were ever accidentally deleted or lost.

5.5.3 Unanticipated risks encountered, attempts to manage and success

An unanticipated risk that the group encountered was the release of a new version of Matlab while the project team was still developing the end product. Unfortunately, this caused problems with virtually every component of the program. The client wanted a project that would be compatible with both the new and previous version and the project team struggled to find methods of zooming and panning the plots that would work on both versions. Solutions were found that allowed all of the features to work on both versions.

5.5.4 Resultant Changes in Risk Management Made Because of Encountered Unanticipated Risks

Because of the need for the project to work well with both versions of Matlab, all releases are tested on both versions to ensure compatibility.

5.6 Project Team Information

Jared Kline, Electrical Engineering, German
422 Stonehaven Dr #13
Ames, IA 50015
Phone: 515-480-2606
Email: jkline@iastate.edu

Mark Reisinger, Electrical Engineering
1831 Fuller Road #18
West Des Moines, IA 50265
Phone: 319-621-4986
Email: bluemonk@iastate.edu

Adam Wroblaski, Electrical Engineering
1106 Pinon Dr #5
Ames, IA 50014
Phone: 515-441-1396
Email: youareme@iastate.edu

Yu Chan, Computer Engineering, Mathematics
315 Welch Ave
Ames, IA 50014
Phone: 515-292-5171 ext 211
Email: ychan@iastate.edu

5.7 Client Information

General Electric Company

Primary contact:

Doug Welsh, Manager - Software Products

Address: 1 River road, Bldg. 2, Rm. 623
Schenectady, NY 12345

Telephone: 518 385 5619

Mobile: 518 281 4341

Email: douglas.welsh@ge.com

Fax: 518 385 3165

Teleconference: 877-653-7365 Pass Code: 191535

Secondary contacts:

Dan Leonard (Machine Testing)

Jaun Sanchez (Software Team)

5.8 Advisor Information

Professor Chen-Ching Liu
1117 Coover Hall
Iowa State University, IA 50014
Phone: 515-294-4763
Email: liu@iastate.edu

5.9 Closing Summary

General Electric engineers currently spend three to five days comparing simulation models to the test results from only one of their generators. This process currently involves the engineer changing a large number of inputs many times in complex text based simulation software. The goal of this project was to create a fast and highly usable program that can simplify and accelerate this process. As has been extensively

shown above, the team has created just such an interface, allowing engineers to import and export data sets from their physical testing, perform a variety of simulations with many time reducing enhancements, and quickly determine the parameter values for the generator. By reducing the time needed to run the simulation, vary the parameters, and compare the results to test data from several days to half a day, the company can save hundreds of thousands of dollars in man hours each year.

Appendix A. Sample Test Data

Time, sec	Vt, pu	Efd, pu	Spd, pu	lfd, pu
0.01	0.838	0.080	1.911961	0
0.02	1.024	1.053	1.912173	0
0.03	1.024	1.343	0.999919	0
0.04	1.024	1.354	1.000052	0
0.05	1.024	1.362	0.999749	0
0.06	1.025	1.370	0.999986	0
0.07	1.023	1.340	1.000011	0
0.08	1.024	1.311	1.000044	0
0.09	1.024	1.336	0.999877	0
0.10	1.024	1.341	0.999898	0
0.11	1.024	1.337	1.000077	0
0.12	1.024	1.321	0.999906	0
0.13	1.024	1.327	0.999855	0
0.14	1.024	1.352	0.999912	0
0.15	1.024	1.351	0.999923	0
0.16	1.024	1.308	0.999853	0
0.17	1.024	1.320	0.999923	0
0.18	1.024	1.324	0.999948	0
0.19	1.024	1.334	0.999877	0
0.20	1.024	1.330	0.999842	0
0.21	1.024	1.315	1.000073	0
0.22	1.024	1.332	1.000003	0
0.23	1.024	1.329	0.999991	0
0.24	1.024	1.358	0.999722	0

Appendix B. Sample GUI Setting File

```
clear parameter_setting;
parameter_setting=struct(...
    'workspace_name',{...
        'tpdo','tppdo','tppgo','h','d','ld','lq',...
        'lpd','lppd','ll','s1','s12','ra','rcomp','xcomp',...
        'Pgen_Vstep','Qgen_Vstep','V1_Vstep',...
        'Pgen_Ld_Reject','Qgen_Ld_Reject','V1_Ld_Reject',...
        'Pgen_Flt','Qgen_Flt','V1_Flt',...
        'Tup_Vstep','Vup_Vstep',...
        'Treject_Ld_Reject',...
        'Tflt_on','Tflt_off','Zfault',...
        'Rt','Xt','Rl','Xl','MVA_machine','MVA_system','fbase',...
        'Tstop','Step_size' ...
    },...
    'min_value',{...
        2, .01, .01, .5, 0, .5, .5,...
        .1, .1, .1, .01, .1, 0, 0, -.1,...
        0, -1000, 0,...
        0, -1000, 0,...
        0, -1000, 0,...
        0, -100,...
        0,...
        0,0,0,...
        0, 0, 0, 0, 1, 1, 0,...
        0, .001 ...
    },...
    'max_value',{...
        15, .1, .1, 5, 5, 2.5, 2.5,...
        .3, .3, .3, .5, 1, .01, .1, .1,...
        1000, 1000, 1.05,...
        1000, 1000, 1.05,...
        1000, 1000, 1.05,...
        10, 100 ,...
        10,...
        10,10,100,...
        10, 10, 10, 10, 1000, 1000, 100,...
        60, .01 ...
    },...
    'default_value',{...
        7, .03, .035, 3, 0, 1.2, .8,...
        .22, .2, .16, .05, .3, 0, 0, 0,...
        100, 32.09, 1.05, ...
        100, 32.09, 1.05, ...
        100, 32.09, 1.05, ...
        1,1,...
        1,...
        1,1.1,0,...
        0, .1, 0, .2, 120, 100, 60,...
        30, .004 ...
    },...
    'previous_value',{0},...
    'current_value',{...
        7, .03, .035, 3, 0, 1.2, .8,...
        .22, .2, .16, .05, .3, 0, 0, 0,...
    })
```

```

100, 32.09, 1.05, ...
100, 32.09, 1.05, ...
100, 32.09, 1.05, ...
1,1,...
1,...
1,1.1,0,...
0, .1, 0, .2, 120, 100, 60,...
30, .004 ...
},...
'display_name',{...
'Tpdo','Tppdo','Tppgo','H','D','Ld','Lq',...
'Lpd','Lppd','Ll','S1','S12','Ra','Rcomp','Xcomp',...
'Pgen_Vstp','Qgen_Vstp','V1_Vstp',...
'Pgen_Rjct','Qgen_Rjct','V1_Rjct',...
'Pgen_Flt','Qgen_Flt','V1_Flt',...
'Tup_Vstep','%V_Vstep',...
'T_Rjct',...
'T_flt_on','T_flt_off','Zfault',...
'Rt','Xt','Rl','Xl','MVA_mach','MVA_sys','fbase',...
'Tstop','Step_Size' ...
},...
'description',{''},...
'color',{...

'yellow','yellow','yellow','yellow','yellow','yellow','yellow',...

'yellow','yellow','yellow','yellow','yellow','yellow','yellow','yellow'
,....
'white','white','white',...
'white','white','white',...
'white','white','white',...
'white','white',...
'white',...
'white','white','white',...
[1 .4 .4],[1 .4 .4],[1 .4 .4],[1 .4 .4],[1 .4 .4],[1 .4 .4],[1
.4 .4],...
[.7 .7 .9],[.7 .7 .9] ...
},...
'slider_panel',{...
1,1,1,1,1,1,1,...
1,1,1,1,1,1,1,1,...
2,2,2,...
2,2,2,...
2,2,2,...
2,2,...
2,...
2,2,2,...
3,3,3,3,3,3,3,...
4,4 ...
},...
'editability',{...
0,0,0,0,0,0,0,...
0,0,0,0,0,0,0,0,...
1,1,1,...
2,2,2,...
3,3,3,...
1,1,...

```

```

%0 - All Tests
%1 - Test 1 Only
%2 - Test 2 Only
%3 - Test 3 Only
%4 - Test 4 Only
% etc

```

```

        2,...
        3,3,3,...
        0,0,0,0,0,0,0,...
        0,0 ...
    }...
);

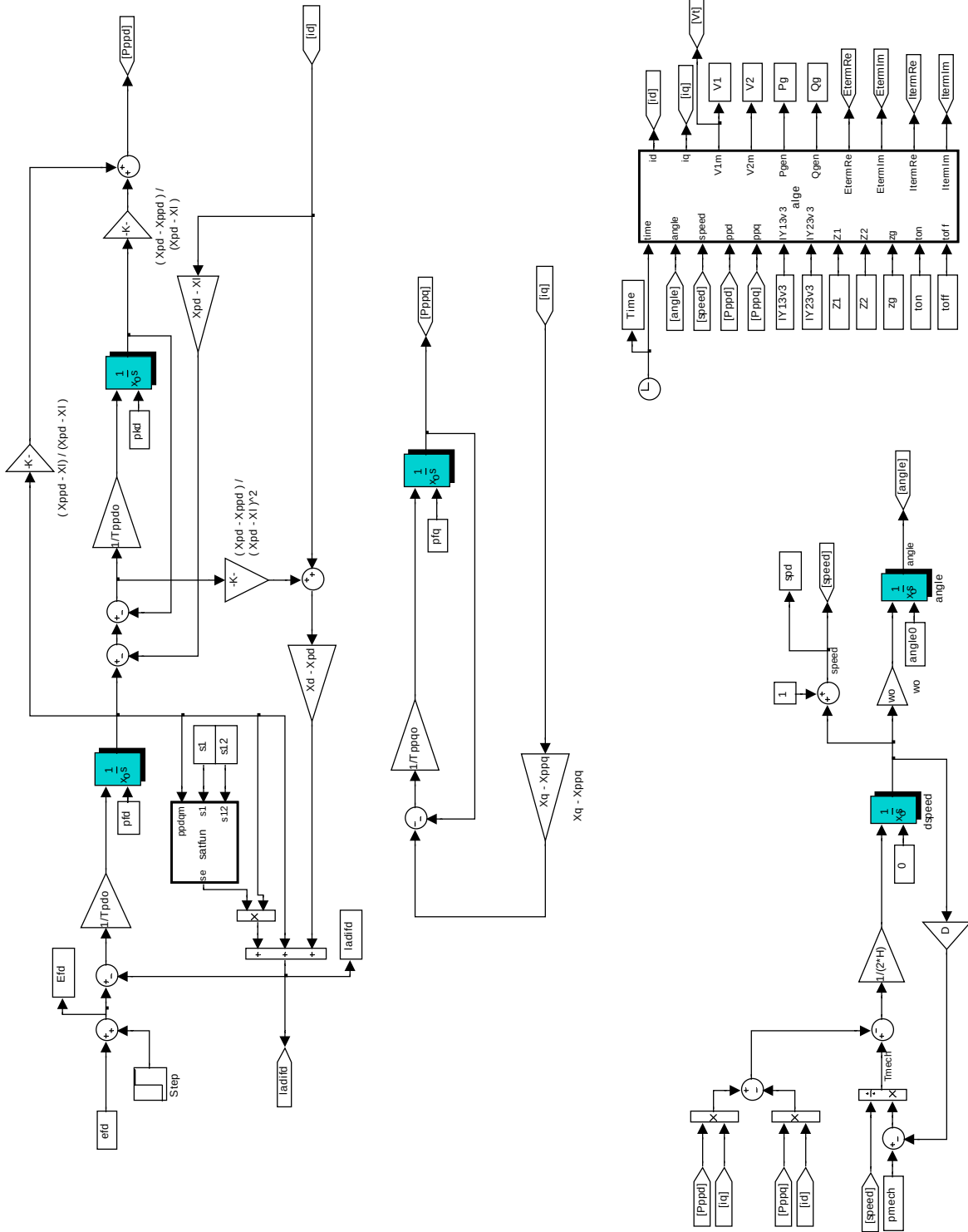
output_setting=struct(...
    'workspace_name',{...
        'V1','Efd','spd','ladifd'...
    },...
    'start_time',{0},...
    'end_time',{10},...
    'display_name',{...
        'V1_t [pu]','Efd [pu]','speed [pu]','ladifd [pu]'...
    },...
    'test_data',{'x'},...
    'simulation_data',{'x'},...
    'old_simulation_data',{'x'},...
    'test_time',{0}...
);

tests_setting=struct(...
    'test_name',{...
        'Voltage Step','Load Rejection','Three Phase Fault'...
    }...
);

simulin_file_name='SMIBgensal_v1';
simulation_file='Gensal Model';
simulation_script='SMIBgensal_sim';

```

Appendix C. Sample Simulink Block Diagram



SMIB
02/15/07

Appendix D. Detailed Flowchart

